UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boris Laharnar

Simulatorji interneta stvari

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJ RAČUNALNIŠTVA IN INFORMATIKE
SMER INFORMATIKA

Ljubljana, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boris Laharnar

Simulatorji interneta stvari

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJ RAČUNALNIŠTVA IN INFORMATIKE
SMER INFORMATIKA

Mentor: prof. dr. Marko Bajec

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Simulatorji interneta stvari

Tematika naloge:

V okviru diplomske naloge naredite obširen pregled IoT simulatorjev. Osredotočite se na simulatorje, ki so prosto dostopni in odprto-kodni. Vsak simulator posebej opišite, za vse skupaj pa pripravite pregledno tabelo, ki bo simulatorje primerjala po ključnih parametrih in bo lahko služila zainteresiranemu raziskovalcu, da za svoje potrebe izbere najbolj primeren simulator. Uporabnost takšne primerjalne tabele prikažite na primeru.

Internet stvari (IoT) je eden izmed najpomembnejših tehnoloških trendov, kateremu mnogi pripisujejo ključno vlogo pri oblikovanju digitalne ekonomije prihodnosti ter razvoju informacijske družbe nasploh. Vlaganja, ki se strmo povečujejo, to potrjujejo, dodana vrednost pa naj bi do leta 2020 po nekaterih ocenah dosegla celo 2 bilijona dolarjev. Investicije v digitalizacijo s pomočjo interneta stvari (npr. pametna mesta, pametne tovarne, pametne stavbe, pametna raba energije, e-zdravje itd.) pa so obenem tvegane, saj težko ocenimo, kdaj se bo investicija povrnila. Zato so zelo dobrodošli IoT simulatorji in platforme, ki omogočajo, da posamezna rešitve, tudi kompleksnejše (kot npr. koncept pametnega mesta) preskusimo najprej v simuliranem okolju in na ta način analiziramo potencialne učinke.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Boris Laharnar, z vpisno številko 24950367, sem avtor diplomskega dela z naslovom:

Simulatorji interneta stvari.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Bajca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela in
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

.

V Ljubljani, dne _____ Podpis avtorja: _____

Zahvaljujem se družini za neverjetno potrpežljivost in enkratno spodbudo.

Zahvaljujem se mentorju prof. dr. Marku Bajcu za strokovno izvedbo mentorstva, promptno pomoč in prijazne nasvete.

Brinu in Špeli.

Kazalo

Povzetek ..... **xix**

Abstract ..... **xix**

1 Uvod **1**

1.1 Dodana vrednost IoT 1
1.2 IoT naprava 2
1.2.1 Senzor 3
1.2.2 Programiranje IoT naprav 4
1.2.2.1 Brezžično senzorsko omrežje (WSN) 4
1.2.2.2 Medmrežni vmesnik (gateway) 4
1.2.3 IoT komunikacija 5
1.2.4 Nizkoenergijski protokoli 6
1.3 Operacijski sistemi za IoT 6
1.4 Vmesno programje (middleware) 7

2 Simulacija IoT okolja **9**

2.1 Dejavniki tveganja IoT projektov 10
2.2 Obvladovanje razvoja IoT s simulacijami 11
2.3 Modeliranje sistema IoT 12
2.4 Rešitve za podporo simulacijam v IoT 13
2.5 Izgradnja ciljnega modela in izbira simulacijskih orodij 15
2.6 Želene lastnosti simulacijske platforme za IoT 16

3 IoT simulatorji **19**

3.1 Opisi simulacijskih orodij in rešitev za IoT 19
3.1.1 Simulatorji 19
3.1.1.1 UbiREAL 19
3.1.1.2 OMNeT++ 21
3.1.1.2.1 INET 23
3.1.1.2.2 MiXiM 24
3.1.1.2.3 Veins 24
3.1.1.2.4 Castalia 24
3.1.1.2.5 Druge razširitve platforme OMNeT++ 25
3.1.1.3 CupCarbon [43] 25
3.1.1.4 Persepteur [44] [45] 27
3.1.1.5 NS-3 29

3.1.1.5.1 iTETRIS (Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions) 30
3.1.1.5.2 NS-2 30
3.1.1.6 EASIM 31
3.1.1.7 SimPy 32
3.1.1.8 Ptolemy II 32
3.1.1.9 Shawn 33
3.1.1.10 SIDnet-SWANS [62] [63] [64] 34
3.1.1.11 GrooveNet [64] 34
3.1.1.12 Worldsens Simulator 35
3.1.1.13 Freedomotic 36
3.1.1.14 Avrora [53] 37
3.1.1.15 StreetLightSim 38
3.1.1.16 DEUS 38
3.1.1.17 OSMobility 39
3.1.1.18 MAMMotH 39
3.1.1.19 DEVSimPy 40
3.1.1.20 ZBOSS Network Simulator 40
3.1.1.21 Automatski 41
3.1.1.22 Desmo-J 41
3.1.1.23 ArduPilot 42
3.1.1.24 Mosaik 42
3.1.1.24.1 Mosaik + OMNeT++ 43
3.1.1.25 TRMSim-WSN (Trust and Reputation Models Simulator for Wireless Sensor Networks) 44
3.1.1.26 Sensor Security Simulator (S3) 44
3.1.1.27 WSN Localization Simulator 45
3.1.2 Simulatorji konteksta (modeliranje agentov) 46
3.1.2.1 Netlogo [9] 46
3.1.2.1.1 Druga orodja na področju modeliranja agentov 47
3.1.2.2 Siafu 47
3.1.2.3 SUMO (Simulation of Urban MObility) 48
3.1.2.4 BonnMotion 49
3.1.3 Virtualizatorji 49
3.1.3.1 DPWSim 50
3.1.3.1.1 DPWS Explorer 51
3.1.3.2 Hue emulator 51
3.1.3.3 AllJoyn Device Simulator 52
3.1.4 Simulatorji senzorjev 53

3.1.4.1 SensorSimulator 53
3.1.4.2 tsdbwriter 53
3.1.5 Simulatorji, emulatorji in virtualizatorji vgradnih naprav 54
3.1.5.1 Atmel AVR 54
3.1.5.1.1 ATEMU (ATmel EMUlator) [53] 54
3.1.5.1.2 SimulAVR [81] 54
3.1.5.1.3 SimAVR 55
3.1.5.1.4 Atmel AVR Simulator 55
3.1.5.1.5 Emulare 55
3.1.5.1.6 Arduino Debugger/Simulator 56
3.1.5.1.7 Simuino 56
3.1.5.1.8 UnoArduSim 56
3.1.5.2 ARM 57
3.1.5.2.1 GNU ARM Eclipse 57
3.1.5.2.2 mbed SDK 57
3.1.5.3 Texas Instruments 57
3.1.5.3.1 MSPSim 57
3.1.5.3.2 WSim 57
3.1.5.4 Android in iPhone 57
3.1.5.5 Drugo 58
3.1.5.5.1 EMUL8 58
3.1.6 Simulatorji omrežnih algoritmov 58
3.1.6.1 Atarraya 58
3.1.6.2 AlgoSenSim 58
3.1.6.3 Sinalgo 59
3.1.7 Operacijski sistemi 59
3.1.7.1 Contiki / Cooja 59
3.1.7.2 RIOT 62
3.1.7.3 TOSSIM 63
3.1.8 Razvojna in prototipna orodja 63
3.1.8.1 Reactive Blocks [90] 63
3.1.8.2 CodeBlocks Arduino IDE 65
3.1.8.3 EvoThings Studio 65
3.1.8.4 WComp 66
3.1.9 Druga orodja za podporo simulacijam 66
3.1.9.1 wotio/ripple 66
3.1.9.2 .NET Gadgeteer 66
3.1.9.3 Automate 67
3.1.9.4 DomoNet 68

3.1.9.4.1 DomoML ... 68
3.1.9.5 Eclipse IoT ... 69
3.1.9.6 Ogrodje za testiranje WoT ... 69
3.1.9.7 EXEHDA-AD ... 69
3.1.9.8 Node-RED ... 69
3.1.10 Raziskovalni projekti in metodologije ... 69
3.1.10.1 Dinamično vizualno simulacijsko orodje za IoT [12] ... 70
3.1.10.2 Metodologija za skalabilne simulacije [29] ... 70
3.1.10.3 DiaSuite ... 71
3.2 Primerjalna tabela med zbranimi orodji ... 71
3.2.1 Tabela I - Osnovna primerjava ... 72
3.2.2 Tabela II - Primerjava po parametrih ... 79
4 Primer uporabe ... 85
5 Zaključki ... 89
Literatura ... 93

Seznam uporabljenih kratic

KRATICA	ANGLEŠKO	SLOVENSKO
ABM	Agent based modelling	Agentno modeliranje
API	Application program interface	Programski vmesnik
BLE	Bluetooth low energy	Nizkoenergijski Bluetooth
CPS	Cyber-physical system	Kiberfizični sistem
DPWS	Devices profile for web services	Profil naprave za spletne storitve
HIL	Hardware in the loop	Vključitev fizične naprave v simulacijo
IIoT	Inudstrial IoT	Industrijski IoT
IoT	Internet of Things	Internet stvari
M2M	Machine to machine	Komunikacija med napravami
MANET	Mobile ad hoc network	Mobilno ad hoc omrežje
MQ	Message queue	Sporočilna vrsta
MWSN	Mobile wireless sensor network	Mobilno senzorsko omrežje
OT	Operational technology	Operacijska tehnologija
OTA	Over-the-air	Brezžična posodobitev
PAN	Personal area network	Osebno omrežje
SIL	Software in the loop	Vključitev druge programske opreme v simulacijo
SOA	Service oriented architecture	Servisno orientirana arhitektura
SVN	Subversion	Repozitorij izvorne kode
TSDB	Time series databases	Časovna podatkovna baza
UAV	Unmanned aerial vehicle	Brezpilotni zrakoplov
V2X	Vehicle to everything	Komunikacija avtomobil - karkoli
VANET	Vehicular ad hoc network	Prometno ad hoc omrežje
WBAN	Wireless body area networks	Brezžično telesno omrežje (omrežja naprav na človekovem telesu)
WPAN	Wireless personal area network	Osebno brezžično omrežje
WSN	Wireless sensor network	Brezžično senzorsko omrežje

Povzetek

Diplomsko delo predstavlja bogat pregled aktualnih prostodostopnih in odprtokodnih simulacijskih orodij za IoT.

Delo obravnava prostodostopne simulatorje ter druga orodja in vire, uporabne za simulacijo IoT primerov uporabe. Opisano je bilo okoli 80 orodij in virov. Med orodji smo navedli simulatorje za domenska področja (pametni dom, povezani promet, pametno mesto, pametno omrežje, UAV...), omrežne simulatorje, simulatorje WSN omrežja, splošne simulatorje diskretnih dogodkov, virtualizatorje naprav, emulatorje in simulatorje vgradnih naprav, simulatorje konteksta (orodja za agentno modeliranje), generatorje podatkov (GPS, MQ), simulatorje senzorjev, kosimulatorje. Vezano na simulacije pa smo obravnavali še: orodja za prototipiranje, operacijske sisteme realnega časa (za naprave z omejenimi viri), vmesno programje, razvojne platforme, javno dostopne testne platforme, metodologije, ontologije in programske specifikacije za izgradnjo samostojnega simulatorja.

Zajeta orodja so opisana s poudarkom na ključnih funkcionalnih karakteristikah. Za lažje pregledovanje in izbiro so posebej predstavljeni še s primerjalno tabelo. Opisani so možni pristopi k modeliranju skupaj s praktičnimi napotki za uporabo primerjalne tabele. Na osnovi praktičnega primera uporabe je prikazan postopek vzpostavitve modela in izvedbe simulacije.

Predstavljena je problematika razvoja IoT rešitev, identificirani so dejavniki tveganja takšnih projektov, kar rešujemo z modeliranjem in simulacijo. Iz dela izhajajo sklepi in ugotovitve: naravna korelacija med simulatorji odprtokodnega tipa in ad hoc arhitekturo IoT, hibridno modeliranje IoT sistema z zlaganjem odprtokodnih simulacijskih orodij po principu lego-kock, uporaba agentnih simulatorjev za humanizacijo simulirane situacije, simulacija kot hrbtenica razvoja IoT.

Ključne besede: IoT, modeliranje, simulacija, odprtokodne rešitve, pametni dom, pametno mesto, pametni prostori, prodorni sistemi, vseprisotno računalništvo, telemedicina, povezani promet, MANET, V2X, vgradne naprave, M2M, WSN, ad hoc omrežja, naprave z omejenimi viri, senzorji, medmrežni vmesnik, storitve oblaka

Abstract

The thesis delivers an extensive review of free and open-source simulation tools useful for IoT simulation. The work deals with free simulators and other tools and sources applicable to simulation of IoT use cases. Roughly 80 tools and other sources were reviewed. Different application domain simulators were enlisted (smart home, connected vehicles, smart city, smart grid, UAV...), network simulators, WSN simulators, discrete event simulators, virtualization tools, emulators and simulators of embedded devices, context simulators (agent based modelling tools), data generators (GPS, MQ), sensor simulators and cosimulators. In

connection with simulation we also listed: prototyping tools, real time operating systems (for resource-constrained devices), middleware, development platforms, open testbeds, metodologies, standalone simulator software specifications.

Enlisted tools were described with emphasis on key functional characteristics. A comparative table was added to facilitate the process of searching for required tools. Possible procedures for modelling are included, as well as practical tips how to use the use of comparative table. The procedure of modelling and execution of simulation was presented by practical use case.

Several risk factors and issues associated with IoT development were identified. A possible approach to deal with them is modelling and simulation. Conclusions and findings: natural correlation between open source simulators and IoT ad hoc architectural characteristics, intuitive modelling by assembling open source simulation tools in manner of lego bricks, usage of agent-based modelling to humanize the simulted situation, simulation as a backbone of IoT development.

Keywords: IoT, modelling, simulation, open-source solutions, smart home, smart city, smart spaces, pervasive system, ubiquitious computing, telemedicine, connected vehicles, MANET, V2X, embedded devices, M2M, WSN, ad hoc networks, resource constrained devices, sensors, gateway, cloud services

1 Uvod

Internet stvari (IoT - Internet of Things) označuje povezovanje vsakdanjih fizičnih objektov v Internet. [1] Računalnikom, tablicam, pametnim telefonom, televizijam se na Internetu pridružijo še fizične entitete: predmeti, stavbe, vozila, živali, osebe. Most med fizičnim in elektronskim svetom vzpostavimo z uporabo senzorjev in aktuatorjev.

Senzorji v IoT so naprave, ki znajo pretvarjati fizikalne, kemične in druge pojave okolja v ustrezne podatke. Poznamo kemične senzorje (kislost, plini), biosenzorje (polimeri, celice), tipala (temperatura, hrup, tlak, vlaga) [2] ter druge senzorske naprave - GPS sprejemnik, kamera, RFID čitalec.

Aktuatorji po drugi strani omogočajo funkcionalnost povratne funkcije vplivanja na okolje. Na daljavo aktiviramo operacije kot so obračanje kamere, aktivacija črpalke, zapiranje lopute, dvig mostu, prikazovanje besedila, premik v prostoru, operacija robotske roke. Aktuatorji so različni elektromehanski elementi - stikala, motorčki, prikazovalniki, zvočniki in podobni gradniki.

Senzorji proizvajajo podatke o stvari (iz okolja), z aktuatorji pa podpremo izvedbeno funkcijo stvari (do okolja), skupaj jih povezujemo v kiberfizične sisteme (CPS). [3]

S tem, ko določeno *stvar* opremimo s smiselno izbranimi senzorji in aktuatorji, jo elektronsko omogočimo. S pomočjo komunikacijske tehnologije gradnike povežemo in preko ustreznih vmesnikov izvedemo podatkovno povezavo na IP omrežje. *Stvar* pridobi unikaten IP naslov in postane sestavni del internetnega ekosistema. Lahko jo iz kjerkoli naslavljamo, upravljamo, programiramo, uporabljamo kot resurs oz. storitev.

1.1 Dodana vrednost IoT

Obstoječi sistemi s področja operativne tehnologije (OT) vključujejo množice senzorjev, preko katerih izvajajo monitoring sistema ter aktuatorjev, preko katerih povratno upravljajo s fizičnimi napravami in drugimi procesi. [4] Senzorji spremljajo stanje okolice in konstantno generirajo masovne podatke (t.j. »big data«). Ti podatki se uporabljajo npr. za tekočo detekcijo okvar z uporabo relativno enostavnih algoritmov. [5] Povezovanje omejenih senzorskih naprav na Internet omogoča dostop do neomejenih procesorskih in pomnilniških resursov (oblaka) in izvedbo naprednejših algoritmov (npr. iz področja umetne inteligence) nad podatki. Vzpostavimo lahko sisteme, s katerimi lahko zaznamo postopne spremembe (npr. spremembe v vibracijskem zapisu motorja) in napovemo določeno okvaro, še preden se ta res pripeti in ustavi delovni proces. Vzpostavljamo lahko sisteme, ki se avtomatsko prilagajajo določenim preferencam (npr. preferencam uporabniku na področju pametnih prostorov). Na oblaku povezujemo podatke skalabilnih populacij milijonov aktivnih senzorjev

in jih uporabljamo v aplikacijah in storitvah nove generacije, vzpostavljamo popolnoma nove uporabniške izkušnje.

Obstoječi sistemi OT so zaprti, licenčni (npr. v industriji SCADA), narekujejo uporabo originalnih nadomestnih delov, podporo s strani pooblaščenih serviserjev in dobaviteljev programske opreme. [6] Povezovanje senzorskih sistemov v Internet spodbuja povezovanje celih sistemov, standardi združljivosti pa omogočajo arhitekturno odprtost, več možnosti razvoja raznolikih aplikacij in posledično več uporabnikov. To pomeni večje proizvodne serije naprav, več povpraševanja po tehnologiji, masovno proizvodnjo komunikacijske strojne opreme, nižje cene na napravo. [7] Proizvodnja strojne opreme ni več patentno omejena na enega proizvajalca. Naprave postanejo medsebojno nadomestljive, sistemi s tem razširljivi, prožni, ugodnejši za vzdrževanje. Standardni API-ji na samih napravah po drugi strani pripomorejo k lažji integraciji in olajšajo aplikativni razvoj. Vse našteto pomeni večkratni prihranek pri vpeljavi, tudi pri nadgradnji obstoječih delujočih sistemov (npr. z vpeljavo ustreznih vmesnikov).

V tej luči so razumljive enormne rasti trga, ki ga napovedujejo analitiki. Leta 2007 je bilo aktivnih 10 milijonov senzorjev po vsem svetu. Leta 2015 smo imeli na račun pametnih telefonov 15 milijard aktivnih senzorjev, do leta 2025 pa naj bi se število povzpelo na 1 bilijon aktivnih senzorjev. [8]

Področja uporabe IoT so zelo različna: industrija, mesta, javna služba, zdravje, potrošnja, promet, notranji prostori, okolje.

1.2 IoT naprava

Internet stvari je distribuiran sistem heterogenih vozlišč - IoT naprav. IoT naprava oddaja določene podatke o stanju (vlaga, temperatura, hitrost vetra) in podpira različne funkcionalne vhode (npr. stikalo v termostatu, motorček za zapiranje okna).

IoT napravo v osnovi sestavljajo:

1. Sklop senzorjev oziroma aktuatorjev.
2. Mikrokrmilnik (minimalna procesorska in pomnilniška funkcija), na katerega so fizično povezani senzorji in aktuatorji.
3. Komunikacijska enota, preko katere se mikrokrmilnik povezuje na IP omrežje (direktno ali pa posredno preko medmrežnega vmesnika - gatewaya).

Naprave so narejene po meri glede na končno aplikacijo.

Najbolj razširjena IoT naprava je pametni telefon. Vsebuje bogat nabor senzorjev, pa tudi aktuatorje (zvočnik, LCD prikazovalnik). Povezuje se na Internet, za komunikacijo uporablja WAN omrežje (GSM, UMTS, LTE), pa tudi LAN (WLAN) in PAN (Bluetooth). Vsebuje relativno močno procesorsko enoto in je le do neke mere varčen sistem. Lahko se uporablja

tudi kot medmrežni vmesnik - preko Bluetooth ali USB priključimo druge naprave - senzorje in aktuatorje.

Poznamo izvedbe IoT naprav, pri kateri so mikrokrmilnik, komunikacijska enota in senzorji zasnovani modularno, jih povezujemo med seboj in gradimo IoT napravo po meri (npr. platforme Arduino, mbed, .NET Gadgeteer).

Komplementarna rešitev je vgradnja mikrokrmilnika, komunikacijske enote in senzorjev na en čip. V tem primeru govorimo o sistemu na čipu (system on a chip - SoC), aplikacije vezane na to izvedbo so smart dust, pametni tekstil - PASTA ipd. [9]

IoT naprave so v praksi artikli, ki jih ljudje nosijo na sebi (ura, nakit), naprave v pametni hiši, npr. brezžično tipalo za plin, pametni hladilnik, klimatska naprava. Na področju pametnih mest so to cestne luči, števci prometa, števci za vodo (elektriko, plin), pametni smetnjaki, semaforji, radarji, merilniki kakovosti zraka, mestni avtobusi, zapornice v garažnih hišah, elektronske ključavnice za kolesa, prometna signalizacija. Na področju industrije mehanske lopute, senzorji zalog, merilniki tresljajev na stroju itd.

1.2.1 Senzor

Senzor je pretvornik [10] fizičnih fenomenov v berljive količine - električno napetost, tok, frekvenco. [11] Stimulus je sprememba v okolju, ki aktivira senzor. Vsak stimulus ima specifične zakonitosti. Lahko gre za zvezen pojav (temperatura) ali diskreten dogodek (zaznava gibanja). Aktivira se lahko npr. periodično (prihod v službo) in po neki določeni porazdelitvi (npr. prihod avtobusa).

V osnovi so senzorji torej [12] analogne naprave, saj spremljajo fizični svet, zato meritve niso brezpogojno zanesljive. Neustrezna meritev na senzorju pomeni slabe podatke, na podlagi katerih se izvede napačno ukrepanje - aktivira lažne alarme in po nepotrebnem izpelje poslovna pravila (npr. obveščanje o poplavi). Če se sistem aktivira, čeprav stimulusa na senzorju ni bilo (npr. prižgana luč namesto gibanje človeka ali napaka na tipalu), govorimo o lažno pozitivnih napakah (false positive). Če se po drugi strani ob dejanskem stimulusu senzor ne aktivira (npr. ne sproži alarma v primeru zastoja dihanja pacienta), govorimo o lažno negativnih napakah (false negative). [13]

Izbira senzorjev je ključnega pomena, odvisna je od zahtev načrtovane aplikacije. Za specifičen tip senzorja (npr. senzor temperature) obstajajo različne izvedbe glede na: območje delovanja, pogostost oddajanja signalov, natančnost, zahteve po kalibraciji. Posledično se cene senzorjev istega tipa tako lahko gibljejo od nekaj deset centov do več tisoč dolarjev na kos. Senzorske izvedbe na pametnih telefonih so npr. poceni, vendar odstopajo tudi za 10-15% glede na referenčno laboratorijsko merilno opremo. [14].

S stališča računalničarja je senzor IoT naprava, ki ob določenih dogodkih proizvaja podatke na izhodu. Idealno je, da vsebuje določen nabor vmesnikov v smislu API, kar olajša

aplikativno uporabo naprave. Podoben nivo abstrakcije pričakujemo tudi od aktuatorjev, ki so pravzaprav kontrolni in podatkovni vhodi IoT sistema. Sistemski integrator bi si verjetno po drugi strani želel, da so naprave čim lažje povezljive (plug & play, service discovery itd.).[13]

1.2.2 Programiranje IoT naprav

IoT naprave so pogosto lahko mobilne, postavljene v nedostopne, odročne lokacije, različno gosto distribuirane po prostoru [15]. V teh primerih nimamo na voljo dovolj zanesljivega vira napajanja, npr. zgolj par AAA baterij, zato je varčnost naprave zelo pomembna. To zagotovimo z omejevanjem procesorskih, pomnilniških in komunikacijskih resursov. Konkretneje z bolj pogostim prehodom naprave v stanje pripravljenosti, nižjo hitrostjo prenosa, nižjo močjo oddajnika, uporabo mikrokrmilnika namesto procesorja ipd.

Takšne IoT naprave opravljajo omejeno funkcijo, pogosto uporabniškega vmesnika ni oz. je minimalen in sodijo v kategorijo **vgradnih sistemov** (embedded systems). Za razvoj takšnih naprav potrebujemo specialna programerska znanja. Zapleteno je odkrivanje in odpravljanje programskih napak. Oddaljena naprava ni vedno fizično dosegljiva (npr. instalacije na visokih stavbah). Tudi ni vedno v dosegu omrežja, v tem primeru med drugim ni možna niti nadgradnja programske opreme (OTA). [12]

1.2.2.1 Brezžično senzorsko omrežje (WSN)

WSN (Wireless Sensor Network, ali bolj pravilno WSAN - Wireless Sensor and Actuator Network) je izvedba IoT, pri kateri distribuirana množica varčnih avtonomnih senzorskih naprav komunicira preko radijske povezave.

Doseg posameznega vozlišča je omejen, komunikacija zato poteka od vozlišč do končnega cilja (ponor, medmrežni vmesnik) na različne načine z uporabo posebnih distribuiranih algoritmov. V praksi se uporabljajo različne komunikacijske topologije, npr. klasične omrežne topologije, pri čemer so vozlišča v prostoru nameščena statično ali npr. specialne mobilne ad hoc topologije, npr. MANET in MWSN.

1.2.2.2 Medmrežni vmesnik (gateway)

Naprava z omejenimi viri (resource constrained device) ima omejene možnosti obdelave proizvedenih podatkov. IoT omogoča povezovanje senzorjev z IP omrežjem in s tem ponuja možnost, da se izvajanje naprednejše programske logike prenese na oblak. Omejitve pri stalni povezavi z oblakom pa so dodatna poraba energije, obremenitev omrežja, zasebnost podatkov, stroški uporabe oblaka, stroški prenosa podatkov itn.

V distribuiranem sistemu uporabljamo posebne računalnike - medmrežne vmesnike - za povezovanje različnih vrst omrežij med seboj, npr. senzorskega omrežja in IP omrežja. [16] Te naprave Imajo bolj zanesljiv vir energije, operirajo z več resursi kot jih premorejo senzorske naprave, gostijo priljubljene operacijske sisteme (npr. Linux). Računalniške

komponente za medmrežni vmesnik so: QorIQ, CloudGate, Reliagate, Kontron M2M, Raspberry Pi [17].

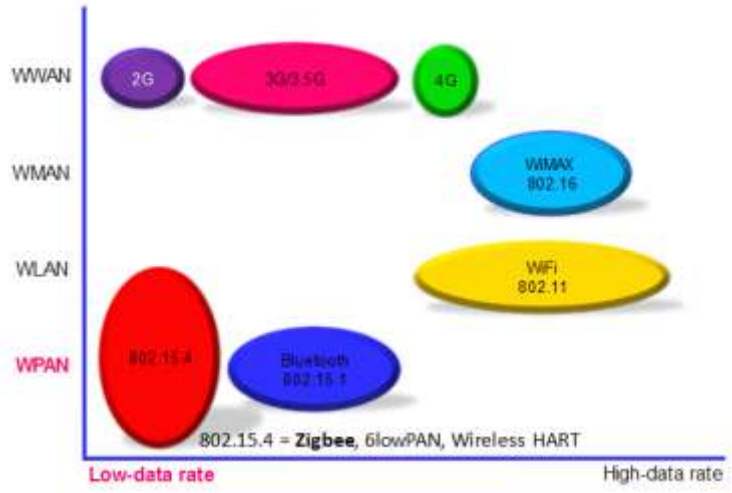
Določen segment programske logike lahko izvajamo tudi na medmrežnem vmesniku. Končna porazdelitev programske logike med vozlišči je odvisna od večih faktorjev kot je vrsta aplikacije, frekvenca nastajanja senzorskih podatkov, stroški podatkovnih prenosov ipd.

1.2.3 IoT komunikacija

IoT naprava beleži stanja v okolju in ki ob različnih dogodkih - npr. ob stimulaciji senzorja, na zahtevo ali periodično - generira podatke in jih pošilja na določeno lokacijo. Primer: inteligentna naprava je nameščena v kabini tovornjaka, spremlja pot in lokacijo ter druge morebitne senzorske podatke in jih periodično sporoča v centralni strežnik. V primeru nesreče, npr. prevračanja tovornjaka pošlje alarm na informacijsko točko, na naprave drugih vozil v bližini itd. Takšno avtomatsko komunikacijo med napravami imenujemo **M2M (machine to machine)**.

Komunikacija M2M je navadno brezžična (radijska). Radijski kanal je izgubni kanal. Različne motnje v komunikaciji povzročajo izgubo paketov, zakasnitve, izpade, zato je potrebno pošiljanje večkrat ponavljati, kar dodatno vpliva na porabo energije, odzivnost in prepustnost. Na kakovost signala med drugim vplivajo tudi ovire kot so stavbe, gostota porazdelitev. [15] Pri tem obstaja vrsta nedoločenosti, ki jih je potrebno rešiti.

Funkcionalne zahteve sicer določajo potreben doseg in hitrost prenosa ter s tem izbiro uporabljenih protokolov. Če je naprava priključena na zanesljiv vir energije (medmrežni vmesnik), varčnost mobilne komunikacije ni pomembna. V tem primeru je bolj enostavno uporabiti splošne brezžične protokole WLAN, Bluetooth, 3G (UMTS), 4G (LTE), WIMAX ali ožičeno povezavo. Če je poraba energije pomembna, uporabljamo prilagojene komunikacijske protokolne sklade za nizkoenergijska omrežja.



Slika 1: Povezava med dosegom brezžičnih tehnologij in hitrostjo prenosa [18]

1.2.4 Nizkoenergijski protokoli

Na področju WAN nastajajo rešitve Low Power Wide Area Network (LPWAN), ki bodo omogočile nizkoenergijsko širokopasovno hrtbtenico za IoT neposredno preko mobilnega operaterja. Posledično bomo lahko izvajali distribucije naprav in M2M komunikacijo na širšem geografskem območju, v stavbah in tudi pod zemljo. Zanimive aplikacije so npr. pametna agronomija (upravljanje kmetije), pametna kolesa, sledenje vozilom, števci porabe vode, sledenje domačim živalim.[19] Rešitve so še v fazi razvoja in testiranja, primer je LoRaWAN.

Na ravni WLAN in WPAN je najbolj razširjen protokolni sklad ZigBee. Uporablja se v napravah v industriji in medicini. [20]. Podatkovni in fizični sloj sta definirana s standardom IEEE 802.15.4, višji sloji pa so licenčni. [21] Na standardu 802.15.4 temelji tudi novejši sklad 6LoWPAN. Je v celoti standardiziran tudi na višjih slojih, podpira IPv6 protokol. Zaradi razširjenosti omenimo še Z-Wave. Gre za licenčno radijsko rešitev, ki se uporablja predvsem v pametnih hišah, tehnologija ni zasnovana večslojno (kot protokolni sklad), za povezovanje z IP omrežjem je potreben vmesnik. [7]. ZigBee in Z-Wave sta zaprti protokolni rešitvi, ki v osnovi nista bili razviti za povezovanje z Internetom, zato v tem primeru potrebujemo medmrežni vmesnik. Licenčni sistemi so neugodni tudi zavoljo licenčnine, ki jo zaračunavajo proizvajalcem strojne opreme. Zasluga za uspeh ZigBee gre zaenkrat še predvsem na račun enostavnejše uporabe od 6LoWPAN. To vrzel pa sicer rešujejo ustrezni operacijski sistemi za IoT, razvojna orodja (IDE) in vmesno programje (middleware).

Na področju WPAN se uporablja tudi protokole na standardu IEEE 802.15.1, npr. Bluetooth Low Energy (BLE), ki se na IP omrežje priključujejo preko vmesnika.

Omenimo še standard IEEE 802.15.6, ki podpira telesna komunikacijska omrežja - komunikacijo naprav zelo kratkega dosega, ki jih ljudje nosijo na sebi (WBAN - Wireless Body Area Network). Aplikacije na tem področju so medicina, zdravje, fitnes.

1.3 Operacijski sistemi za IoT

Za uporabo resursov radijsko povezanih vgradnih naprav z omejenimi viri se uporabljajo **operacijski sistemi realnega časa (RTOS)**, npr. RIOT, TinyOS, Contiki, mbed OS, Zephyr, FreeRTOS. Ponavadi so to 8-bitni sistemi z naborom protokolov za varčno brezžično komunikacijo v PAN in LAN področju. Ti operacijski sistemi torej tečejo na končnih napravah in vsebujejo ves potrebni nabor funkcij OS (razvrščanje poslov, programski dogodki, protokoli). Med delovanjem zasedajo ekstremno malo pomnilniškega prostora (nekaj kilobytov) in olajšajo izvajanje aplikacij na IoT napravi.

1.4 Vmesno programje (middleware)

V distribuiranem sistemu kot je IoT, imamo opravka z različnimi komponentami, aplikacijami, storitvami in drugimi viri, ki so si strojno ali programsko zelo različni, proizvajajo raznolike podatke.

Na nekem nivoju abstrakcije arhitekturnega modela je ugodno predstaviti vse vire kot »storitve«. Vpeljemo principe storitveno orientirane arhitekture (SOA) in upravljanje s podatki iz heterogenih porazdeljenih virov po poenoteni metodi. Ta nivo integracije je podprt z vmesnim programjem (middleware), ki podpira tehnike in mehanizme za povezovanje IoT naprav različnih proizvajalcev v enoten sistem.

2 Simulacija IoT okolja

Razvoj IoT je multidisciplinaren proces - pri izgradnji sodeluje več inženirskih disciplin: elektrotehniki (električna vezja, kontrolniki, protokoli, integracija, vgradno programiranje), informatiki (aplikativni razvoj, integracija, oblak), fiziki (razvoj senzorjev) in drugi strokovnjaki iz področja OT. Višji začetni stroški gredo tudi na račun nabave ustrezne strojne opreme. [12] V določenih scenarijih se uporabljajo tudi senzorji, ki jih ni enostavno testirati v realnem okolju (npr. poplava, požar). [11]

IoT sistemi so pogosto skalabilni - npr. aplikacije v pametnih mestih vključujejo na milijone aktivnih vozlišč. Do leta 2020 napovedujejo 20.8 milijard naprav povezanih v internet. [22] Vsaka naprava proizvaja določeno količino podatkov, ki se pretaka po omrežju in to pomeni velik pritisk na infrastrukturo. [23] Naprave v kritičnih sistemih v realnem času morajo biti natančne, odzivne, zato zagotavljati dobro razmerje uspešno poslanih paketov. Pri veliki populaciji uporabnikov je kritična po eni strani zanesljivost delovanja sistema in po drugi enostavnost uporabe (npr. pri IoT izdelkih široke potrošnje) ali celo nezaznavnost, npr. v primeru prodornih sistemov (pervasive systems).

IoT lahko vključuje mobilnost. Za ta namen so potrebni posebni protokoli. Topologija mobilnega omrežja se avtomatsko prilagaja, za kar se uporabljajo posebni algoritmi za distribuirana omrežja.

Vključevanje naprav v IP omrežje pomeni tudi potencialne vdore, zlorabe, odtujitve podatkov. Vzpostavitev varne komunikacije (šifriranje) pomeni dodatne potrebe po procesorski moči in s tem energiji na varčnih napravah.

Prenos podatkov od naprav do oblaka pomeni fiksne in variabilne stroške prenosa. Obstaja več alternativ izbire komunikacijske poti. Mobilni operaterji ponujajo različne podatkovne plane za M2M, samo delovanje je možno bolje porazdeliti med napravami, oblakom in medmrežnim vmesnikom ipd.

Pri stroških upravljanja je potrebno prišteti še stroške uporabe storitev na oblaku, to vključujejo množico kombiniranih spremenljivk - količina alociranih resursov, število instanc, število priključenih naprav, količina pretečenih podatkov, velikost podatkovne baze itd. Modeli zaračunavanja so kompleksni in variirajo od ponudnika do ponudnika.

IoT podpira adaptivne socio-tehnološke sisteme kot je npr. poslovno okolje, javni prostor, mesto. [24] Mnoge značilnosti v okolju so nepredvidljive in neobvladljive - okolje se obnaša v skladu s socialnimi, psihološkimi in fizikalnimi zakonitostmi.

Za postavitev naprav na trg so potrebni tudi določeni certifikati vezani na lokalno regulacijo (npr. ustreznost radijskega signala / vpliv na zdravje itd.) [24] Uporabniški vmesniki

globalnih produktov pa morajo biti podprti tudi z ustrezno lokalizacijo in internacionalizacijo (formati, črke ipd.). [25]

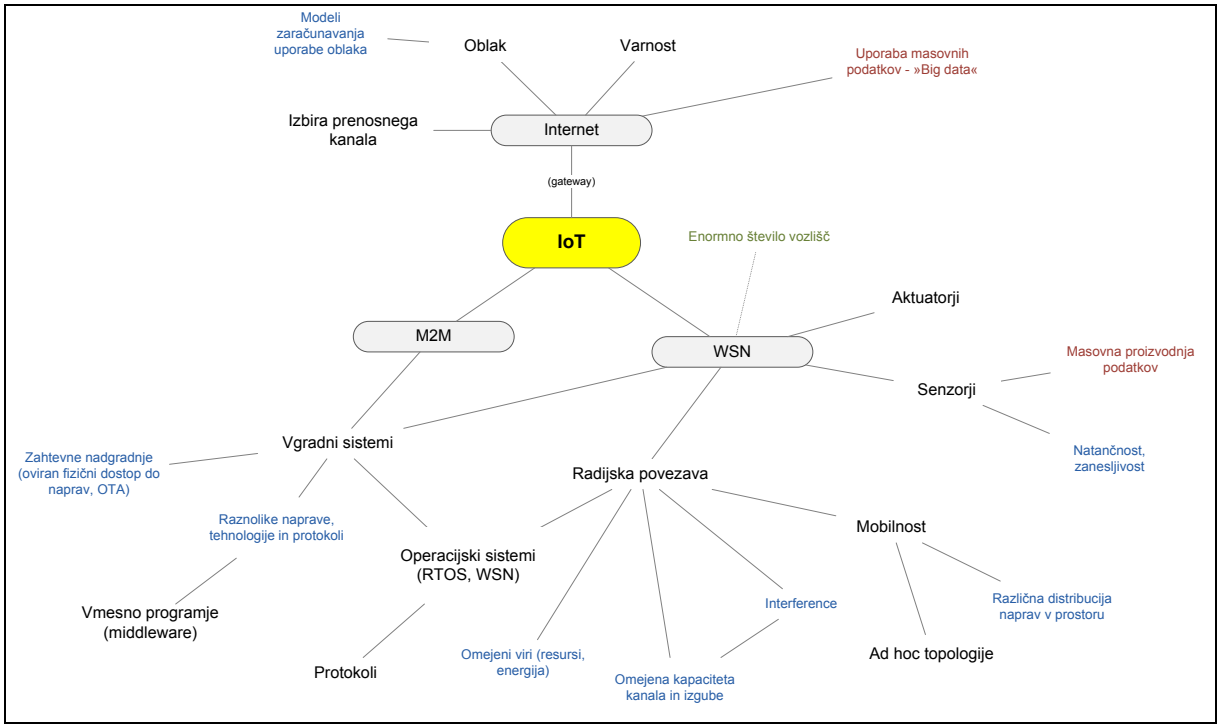
Zelo pomemben vidik je poraba energije. Na to vpliva frekvenca obdelave, mesto procesiranja in izvedba komunikacije. Energetski modeli niso pomembni samo v smislu učinkovite izrabe resursov na mobilnih napravah, ampak tudi delovanja celotnega IoT ekosistema. V letu 2010 je IKT predstavljala 10% porabe celotne energije. Do leta 2030 naj bi ta delež znašal več kot 50%. [26]

2.1 Dejavniki tveganja IoT projektov

Evidentiramo lahko naslednje dejavnike tveganja (tehnološki faktorji / vidiki nedoločenosti) razvoja IoT sistema:

- aplikativna raznovrstnost (od pralnih strojev do merilcev tlaka in števcev prometa),
- raznolikost tehnologij in protokolov (odvisno od dosega in vira energije, veliko je proizvajalcev, obstojajo odprte in licenčne rešitve),
    - razpoložljivost tehnologije (potreba po lastnem razvoju naprav, protokolov),
- skalabilnost aktivnih vozlišč (breme za komunikacijsko infrastrukturo),
- distribucija (gostota) naprav v prostoru (vpliv na zdravje, interferenca, doseg),
- performanse sistema (odzivnost, kapaciteta kanala / zanesljivost prenosa paketov, nenadne spremembe kakovosti signala),
- posebnosti razvoja programske opreme (vgradne naprave),
- procesorske in pomnilniške zmogljivosti na napravah (omejeni viri),
- poraba energije, avtonomija, vir energije (pridobivanje energije - sonce, veter, radiacija, vibracija, termalna energija) - HAEC (Highly Adaptive Energy-efficient Computing),
- mobilnost naprav in dinamične topologije,
- posebnosti nadgradenj (OTA, fizična dostopnost do naprav),
- nezanesljivost delovanja senzorjev (kalibracija, območje delovanja),
- porazdelitev izvajanja logike (naprava, vmesniki, oblak),
- količina generiranih podatkov,
    - stroški prenosa podatkov (podatkovni plani),
    - stroški uporabe oblaka (kompleksni ceniki),
- varnost (šifriranje podatkov, avtentikacija),
- zagonski vložki v infrastrukturo,
- drugi najrazličnejši dejavniki okolja (psihološki, sociološki...),

- različne geografske lokacije (časovni pasovi)



Slika 2: Heterogenost sistema IoT

Ugotavljamo, da je IoT kompleksen sistem heterogenih elektronskih naprav. IoT imenujejo tudi kot »sistem sistemov« ali celo »sistemi sistemov«. [24]

2.2 Obvladovanje razvoja IoT s simulacijami

Strošek za odpravo napake v fazi uvajanja je 29 do 1500 krat višji kot v fazi zajema zahtev [27]. Posebej za heterogene sisteme kot je IoT napake **na projektih** izraziteje eskalirajo. Razvoj IoT vsebuje veliko mero nedoločenosti na večih nivojih, ki jih je potrebno odpraviti čim bolj zgodaj v razvojnem procesu.

Evaluacijo delovanja sistema lahko sicer izvajamo na prototipnih fizičnih napravah. Naprave namestimo v določeno fizično okolje ali laboratorij in preizkušamo konfiguracije in izvedbo rešitve neposredno na njih. Ta pristop ima svoje omejitve. [12] Nakup naprav ni zanemarljiv strošek, predvsem če gre za večji nabor naprav različnih proizvajalcev, med katerimi izbiramo. Programiranje na napravah je bolj zamudno, predvsem instalacije. [28] Naprave so postavljene v določeno kontrolirano okolje, dejavniki potem niso realni, okolje ni reprezentativno. Predvsem pri IoT se srečujemo s populacijami, ki obsegajo na stotisoče aktivnih vozlišč in laboratorijska testna okolja ne zadoščajo za ustrezno obravnavo sistemov takšnega obsega. [29]

IoT je kompleksen sistem, ki zahteva temeljito in tehtno analitično obravnavo. Simulacija omogoča evaluacijo IoT rešitve na scenarijih simuliranega okolja. S simulacijami preverjamo omejitve sistema (npr. performančne), analiziramo nevarne scenarije in neželene situacije. V specifičnem stanju okolja primerjamo izbiro algoritmov, strojnih komponent, aplikativnih izvedb.

Značilno za vgradne sisteme je, da strojna oprema pridobi končno podobo šele ob koncu razvojnega projekta. Ustrezno podprta simulacija v sklopu razvoja je zato ključna. V praksi izvajamo tudi takšne evaluacije, ko strojne naprave fizično priključimo na simulacijsko platformo. Izvajanje logike prepustimo fizični napravi, na kateri predhodno namestimo potrebno programsko opremo, simuliramo pa pogoje okolja, kjer bodo naprave nameščene.

Po mnenju IBM je »modeliranje in simulacija IoT ključni sestavni del procesa izgradnje povezanih produktov v IoT.« [24] Pri IoT gre za nov nivo kompleksnosti in nove dimenzije nedoločenosti v primerjavi z dosedanjimi IT projekti (npr. distribuiranimi sistemi) in simulacija je nepogrešljiv način obvladovanja tveganja.

2.3 Modeliranje sistema IoT

Simulacija je izvedba modelov na računalniškem sistemu. [30] Model je matematična ali drugačna abstraktna podoba realnega stanja. Modeliramo lahko bodisi računalniške komponente (strojna oprema, programska logika, storitve) ali okolje (stimuluse agentov, okoljske dejavnike, omejitve prostora).

Za izgradnjo modelov lahko uporabimo različne nivoje realističnosti:

- dejanske realne izvore podatkov (spletni servisi, API-ji, fizične naprave) - realistični resursi, ki nastopajo tudi v produkcijski aplikaciji,
- emulatorje - elektronske naprave (senzorje, aktuatorje, mikrokrmilnike) implementiramo programsko na način, da dosežemo enakovredno izvajanje programske logike kot na fizični napravi,
- simulatorje (matematične modele) - nedeterministične lastnosti sistema (stimulusi iz okolja, komunikacijski model, energetski model) aproksimiramo z uporabo matematičnih funkcij in algoritmov.

Implementacija modela ni trivialna. [31] Za merodajno simulacijo potrebujemo soliden nabor realnih predpostavk (robnih pogojev), reprezentativno aproksimacijo stanja. Ker opazujemo le določen spekter delovanja sistema, je potrebno dobro premisliti, katere segmente je smiselno natančneje modelirati. Natančni modeli zahtevajo več računanja, kar upočasnjuje izvajanje simulacije. Za dano področje obravnave torej iščemo smiselni model, ki bi bil dovolj realističen, obenem pa bi se izvajal dovolj hitro.

2.4 Rešitve za podporo simulacijam v IoT

Aplikacijske zahteve determinirajo arhitekturo sistema. Rešitve za osebno zdravje vključujejo drugačen nabor naprav, komunikacijskih protokolov, drugačno konfiguracijo oblaka in varnostne politike kot npr. IoT na področju upravljanja oskrbovalne verige v industrijskem IoT (IIoT) ali pametnega mesta.

Po drugi strani bi si želeli obravnavati zgolj nekatere ožje nivoje uporabljene tehnologije, v katerih obstaja nedoločenost. Osredotočili bi se npr. samo na izgubnost komunikacijskega kanala ali evaluacijo specifičnega algoritma, ostale segmente sistema pa bi abstrahirali oz. kompenzirali z aproksimiranimi modeli.

V prvem primeru se orodja segmentirajo horizontalno po **domenah IoT**, v drugem primeru pa vertikalno po opazovanih sistemskih **nivojih IoT.** To sta sicer tudi osnovna parametra, po katerih bomo tudi primerjali zajeta orodja.

V delu smo naleteli na različne simulacijske rešitve za IoT:

1. Domenski simulatorji pokrivajo določeno domeno IoT (pametna hiša, pametno mesto). Vsebujejo prilagojen nabor rutin in metod. Praviloma je podprt grafični vmesnik, kar olajša modeliranje, izvedba simulacije je podprta z vizualizacijo - animacija izvedbe omogoča boljše razumevanje modela.
2. Splošni simulatorji diskretnih dogodkov (v nadaljevanju splošni simulatorji) vsebujejo generični nabor funkcionalnosti - porazdelitev, metod in rutin. Vizualizacija simulacije ni posebej grafično podprta in je praviloma tekstovna. Ti simulatorji so smiselni za modeliranje kompleksnih primerov uporabe na vsebinskem nivoju, pa tudi npr. dejavnikov okolja (stimuluse senzorjev).
3. Emulatorji omogočajo enakovredno izvedbo programske logike na PC kot na dejanski fizični (vgradni) napravi in smo jih omenili v podpoglavju 2.3. Programska koda, ki se izvaja na emuliranih napravah, je identična tisti na dejanskih napravah. Emulacija preseli razvoj iz naprave na PC, s tem omogoči boljši pregled in nadzor nad izvajanjem in pohitri razvoj za vgradne naprave. [32]. Princip emulacije v IoT izvira iz področja razvoja vgradnih sistemov. [12]
4. Simulatorji agentov izvajajo modele naravnega vedenja uporabnikov, psihologije množice, dinamike prometa, psihosocialne stike na delovnem mestu, delovni proces v proizvodnji, kmetijstvu. V model vpeljemo enega ali več vrst agentov, med katerimi poteka interakcija. Ta orodja poimenujejo tudi simulacija konteksta [33], agentno modeliranje (ABM - agent based modelling). V praksi jih povezujemo s prej omenjenimi simulatorji, da ustvarimo boljši model realnega stanja (npr. modeliramo uporabnike, mobilne situacije) v grafično podprti situaciji.

5. Virtualizatorji - simulirana naprava / storitev, ki se v omrežju pojavi kot dejanska fizična naprava / storitev, ima podprte enake vmesnike ipd., npr. virtualizatorji uPnP naprav, REST API-jev.
6. Generatorji podatkov - generirajo podatke, ki jih npr. lahko uporabimo na vhodu drugih simulatorjev, npr. podatke o geolokaciji, senzorske vrednosti, drugi M2M podatki (MQ).
7. Javno dostopne testne platforme (testbeds) so fizične postavitve IoT, ki jih različne raziskovalne institucije priključijo na Internet in ponudijo širši strokovni javnosti na uporabo za namen raziskovanja. Velik evropski projekt je FIT IoT Lab, ki ponuja 2700 različnih naprav instaliranih na različnih lokacijah po Franciji [34]. Drugi so še Living Labs, Smart Cities [12], WISEBED, Kansei [35], Indriya in številni drugi projekti.

Našteta orodja se uporabljajo samostojna, nekatera ponujajo možnost povezovanja (npr. preko TCP vrat), t.i. hibridne platforme, o katerih bomo govorili v naslednjem poglavju.

Simulatorji se sicer pojavljajo v povezavi z naslednjimi sistemi:

1. Operacijski sistemi za vgradne naprave. Opisovali smo jih v podpoglavju 1.3. Podpirajo različne proizvajalce vgradnih strojnih platform, sistemi torej »razumejo« kodo na končnih napravah. Izvajajo programsko opremo prevedeno iz višjih programskih jezikov, npr. C/C++, zato jedra operacijskih sistemov razširjajo v simulacijske platforme za PC (npr. TinyOS v TOSSIM ali Avrora, Contiki v Cooja).
2. Vmesno programje (middleware) smo opisali v podpoglavju 1.4. Nekatere domenske simulacijske platforme podpirajo specifične fizične naprave (npr. termostat določenega tipa), zato se funkcionalnost simulatorjev lahko razširja tudi na middleware področje (npr. Fredomotic).
3. Orodja za prototipiranje. Včasih avtorji namenski simulator označijo kot orodje za prototipiranje, saj takšno orodje v osnovi simulira okolje in omogoča evaluacijo določene distribuirane aplikacijske rešitve, ki je v bistvu prototip. Prototipi so po drugi strani modeli končnega stanja, model pa je sredstvo simulacije. Tudi zato so orodja za prototipiranje zanimiva za našo obravnavo.
4. Razvojne platforme. Razvoj neposredno na napravah ima določene slabosti, poleg stroškov nakupa naprav tudi stroške nameščanja programske kode na naprave (flashing), omejeno razhroščevanje in logiranje napak, zato tudi v tem segmentu srečujemo funkcionalno podporo simulaciji in prototipiranju kot tudi nameščanju programske logike na končne naprave.

Značilno za vgradne sisteme je, da je strojna oprema v kompletni podobi šele ob koncu razvojnega projekta. Ustrezno podprta simulacija oziroma emulacija v fazi razvoja je zato

ključna. V praksi izvajamo tudi takšne evaluacije, ko strojne naprave fizično priključimo na simulacijsko platformo. Izvajanje logike prepustimo fizični napravi, na kateri predhodno namestimo potrebno programsko opremo, simuliramo pa pogoje okolja, kjer bodo naprave nameščene. Vključevanje fizične strojne opreme v model je *hardware-in-the-loop (HIL)*. Če v model vključimo npr. emulacijo naprave, temu pristopu pravimo *software-in-the-loop* (SIL). [36]

2.5 Izgradnja ciljnega modela in izbira simulacijskih orodij

Radi bi modelirali določeno domeno, odsimulirali primer uporabe. Kako se simulacije lotiti?

Za vsebinske simulacije v zgodnjih fazah razvoja, npr. za modeliranje primerov uporabe s področja, ni smiselno vpletati preveč tehnoloških podrobnosti, zato poskušamo poiskati **domenski simulator**. Na voljo so simulatorji za pametno mesto, pametne prostore, pametni promet ipd. Ker so rezultati simulacij uporabni za nadaljnje razvojne faze, je prednost, če so upoštevani standardi, ontologije, ali npr. podpora namestitev na naprave ipd. Uporabni viri za izgradnjo IoT modelov so sicer tudi določene **metodologije**, ki definirajo specifične postopke za izgradnjo modelov za specifično IoT področje. Npr. za pametna mesta, uporabo točno določenih simulacijskih orodij, od katerih vsak podpira ožji nivo IoT sistema. [29]

Če ustrezen domenski simulator ni na voljo, so za modeliranje primerov uporabe primerni tudi **splošni simulatorji**. Pri splošnih simulatorjih ni omejitev glede domene, zato lahko na relativno enostaven način modeliramo tudi primere uporabe, ki zajemajo tudi več IoT domen v eni simulaciji (npr. pametni dom, fitness IoT, pametno mesto). Zgradimo torej čisti primer uporabe na aplikativnem nivoju na simulatorju diskretnega časa brez vpletanja tehnologij in podrobnosti konteksta.

Za natančnejše prototipiranje distribuiranih aplikacij s področij M2M in WSN se uporabljajo **omrežni simulatorji.** Ta orodja sicer privzeto služijo evaluaciji topologij in protokolov, vendar podpirajo postavitve vozlišč (v našem kontekstu IoT naprav), na katerih teče programska logika, med vozlišči pa je vzpostavljena povezava. Vozlišča se odzivajo na stimuluse in sodelujejo pri porazdeljeni izvedbi. Pogosto so dodani tudi različni dodatni modeli, npr. model mobilnosti (vozlišča vsebujejo lokacijsko informacijo in se premikajo po prostoru), energetski model (za evaluacijo porabe energije naprav), radijski model (evaluacija učinkovitosti radijskega signala), model okolja (model stimulusov). S to vrsto simulatorja pravzaprav lahko modeliramo različne domene IoT (npr. OMNeT++), nekateri pa so definirani na ožjo domeno (npr. CupCarbon za pametno mesto).

Zaradi raznolikosti sistema IoT ni vsestranskega simulatorja, ki bi podpiral vse vidike distribuiranega IoT sistema. Smiselna je uporaba specializiranih simulatorjev za posamezne nivoje obravnave, ki jih je možno povezovati (npr. preko TCP vrat) v poljubne konfiguracije. Različne simulacijske rešitve in vire lahko preko API-jev povezujemo med seboj v namenske

hibridne modele. [29] Npr. model zgrajen na **omrežnem simulatorju** povežemo s **simulatorjem agentov** (npr. pozicije ljudi na javnem prostoru, ki nosijo določeno IoT napravo - vozlišče). S tem vzpostavimo sistem, ki ga stimulirajo psihofizični agenti - multiagentni IoT sistem. Na modelu, ki vsebuje zakonitosti množice agentov, lahko evaluiramo uporabo umetne inteligence, npr. adaptivne algoritme in aplikacije. V model lahko povežemo tudi **obstoječe storitve** [11] (poizvedba za trenutno vremensko napoved), **fizične naprave** (telematska naprava mestnega avtobusa priključena na PC), **virtualizatorje** (virtualna senzorska naprava za merjenje onesnaženosti zraka), pa tudi **splošne simulatorje** (za modeliranje porazdelitev poljubnih dejavnikov). S tem pristopom pridobimo zelo realistične hibridne simulatorje v vsebinskem kontekstu.

S povezovanjem pa dobimo hibridno simulacijsko postavitev. Za koordinirano izvajanje se v tem primeru uporabljajo **kosimulatorji**. V hibridnih postavitvah bi si želeli tudi podporo za vmesno shranjevanje stanja fizičnih naprav (checkpoint) za možnost večkratne ponovitve poskusov za iste simulirane situacije. [29] Upoštevati je potrebno tudi to, da fizične naprave (in storitve) tečejo v realnem času, čas simulatorja pa ne nujno, ker je lahko tudi upočasnjen ali pospešen, v tem primeru je potrebno zagotoviti ustrezno sinhronizacijo.

Takšen hibriden model lahko v končni fazi sklenemo s **simulatorjem oblaka**. Na tem področju je kar nekaj orodij, ki omogočajo stroškovno evaluacijo kompliciranih cenovnih shem oblaka in podobno. Simulatorjev za področje oblaka nismo posebej obravnavali, saj bi s tem odprli preveliko temo, navedli bi samo vir [37] zbranih orodij s tega področja.

Dobra simulacijska arhitektura je zasnovana modularno in jo je enostavno razširjati (npr. podpira arhitekturo vtičnikov). Odprtokodne rešitve po drugi strani omogočajo neomejene dograditve in predelave. V tem primeru izberemo platformo, ki je npr. dovolj sorodna ciljnim potrebam in izvedemo dograditev. Možna je tudi semantična prevedba modela obstoječega simulatorja iz druge domene (npr. EASIM).

Za simuliranje izvedb na simulirani vgradni napravi izberemo razvojno orodje s podprto emulacijo ali samostojni emulator. Proizvajalci sicer ponavadi sami zagotovijo SDK-je za programiranje naprav, obstajajo pa tudi prostokodni emulatorji za bolj razširjene strojne module, npr. Arduino.

Če nobena od naštetih strategij ne ustreza, je seveda možnost **izgradnja novega simulatorja**.

Pri razvoju in nadgradnjah simulatorjev si lahko pomagamo z dostopnimi specifikacijami simulatorjev za želeno področje, npr. naleteli smo na specifikacije simulatorjev za pametne prostore [12] ali železniški IoT [29].

2.6 Želene lastnosti simulacijske platforme za IoT

Na osnovi napisanega so želene značilnosti simulatorjev za IoT naslednje:

- intuitivna grafična podpora (za obravnavani segment), tako za modeliranje kot za izvedbo (2D - tloris/ naris stavbe, projekcija mesta, 3D - navidezna resničnost);
- enostavna uporaba sistema, tako za razvijalce modela kot za uporabnike simulacije na vseh inženirskih nivojih (OT, IT,...);
- simulacijo je možno ponoviti večkrat ob istih parametrih oz. nadaljevati od specifične točke dalje; [11] iz česar sledi razhroščevanje modela med izvajanjem simulacije;
- ponovna uporaba kode - uporaba programskega jezika, ki se sicer uporablja na končnih napravah (Contiki) ali prevajanje modela programske logike v kodo naprav (Diasim);
- podpora hibridnim konfiguracijam - povezovanje zunanjih servisov / API-jev, zunanjih naprav, sintetičnih generatorjev podatkovnega toka in drugih zunanjih virov, s čimer poenostavimo modeliranje in pridobimo na realističnosti izvedbe [12] [11] [29];
- vključevanje poljubnih lokacijskih storitev in drugih generatorjev konteksta v model simulacije (na vhod modela pripeljemo simulator GPS);
- časovna kontrola izvajanja (pospešitev, upočasnitev);
- spreminjanje konfiguracije v realnem času (npr. dodajanje, ugašanje in parametriranje naprav med izvajanjem simulacije brez potrebe po ponovnem zagonu simulacije) [12] [29];
- modularna večnivojska programska arhitektura [11] - simulacijsko orodje mora biti odprto za dograditve in prilagoditve glede na specifične zahteve situacije;
- preprosto vključevanje novih naprav (discovery kot npr. bluetooth) [12] [11];
- podpira modeliranje čim širšega nabora scenarijev [11] (op. to se sicer izključuje s podprto vizualizacijo - bolj ko je vizualizacija specifična, manj raznolikih IoT situacij podpira);
- porazdelitev izvajanja na več računalnikov (npr. v primeru simulacij velikega števila vozlišč se breme porazdeli po večih računalnikih);
- emulatorje raznolikih naprav in proizvajalcev (senzorji, RFID, telematske naprave);
- beleženje podrobnosti o izvajanju simulacije (logiranje).

To so nekateri od tehničnih parametrov, po katerih bomo izvedli primerjavo med različnimi orodji za simulacije IoT.

3 IoT simulatorji

IoT je kompleksen porazdeljeni sistem, ki ga spremljajo zelo različni dejavniki tveganja, ki smo jih navedli v poglavju 2.1. Šele v produkcijskem okolju pri polni obremenitvi se razjasnijo tudi skrite omejitve in slabosti sistema. Odprava napak na takšnih sistemih pa je lahko zamudna, zahtevna in draga. Postavitev primerljivega skalabilnega testnega okolja za potrebe tekočega razvoja ni izvedljiva. Da bi nedoločenosti odpravili že v zgodnjih razvojnih fazah, potrebujemo modele in simulacijska orodja..

Obstajajo zelo različna orodja (platforme, ogrodja, okolja, metodologije - v nadaljevanju orodja) za modeliranje in simulacije v IoT. Uporabna so v različnih fazah razvoja, za različne segmente sistema, za različne domene uporabe. Ta orodja bomo pregledali. Kot smo že omenili, so na raziskovalnem področju bolj primerna orodja, ki so odprta za dodelave in nadgradnje z namenom, da lahko določeno platformo po svoje prilagodimo. Posledično se bomo pri obravnavi omejili na prostodostopna in predvsem odprtokodna orodja, prednost pri tem bomo dali novejšim in živim vzdrževanim orodjem.

Orodja so smiselno kategorizirana. Kakšno orodje spada v več kategorij hkrati (npr. vgradne naprave in hkrati WSN), kategorizacijo smo v takšnih primerih izvedli po ključni značilnosti, ki je po našem mnenju najbolj karakterna za orodje.

Obravnavo bi radi predstavili pregledno, strukturirano, da bi strokovnjakom iz različnih panog kar najbolj olajšali izbiro ustreznih simulacijskih orodij za različne potrebe izgradnje in raziskovanja sistema IoT, zato bomo opisni del sklenili s primerjavami med orodij v primerjalni tabeli.

3.1 Opisi simulacijskih orodij in rešitev za IoT

V tem poglavju bomo navedli in podrobneje opisali simulacijska orodja (in druge rešitve v podporo simulacijam v IoT). Najdene rešitve bomo smiselno kategorizirali.

3.1.1 Simulatorji

V tem poglavju bomo navedli in opisali simulatorje .

3.1.1.1 UbiREAL

Platforma: java
Ustanova: Ubiquitous Computing Systems Laboratory (UBI-Lab) - NAIST (Nara Institute of Science and Technology)
Tip orodja: simulator in orodje za prototipiranje pametnega prostora
Namen: vseprisotno računalništvo, pametni prostori, vizualizacija v 3D
Tip licence: odprta licenca, koda ni objavljena
Zadnja verzija: 1.0 (28.9.2012)

Spletna stran: http://ubireal.org/

UbiREAL [38] omogoča zasnovo in izvedbo modela pametnega prostora v 3D okolju. Modeliranje okolja je podprto z grafičnim vmesnikom. V model je možno vključiti različne osebe in naprave kot npr. televizijski sprejemnik, klimatska naprava, svetilke. Naprave so omrežno povezane in spreminjajo svoje stanje glede na kontekst (lokacija določene osebe, temperatura, vlaga, svetloba, hrup).

Fizikalne spremembe okolice so podprte s posebnim simulacijskim modulom, ki posnema zvezne spremembe.

Slika 3: UbiREAL (vir: http://ubireal.org/)

Podprto je izvajanje logike na napravah. To naj bi bilo enakovredno kot na fizičnih napravah, kar pomeni, da gre za emulacijo. Možno je tudi vključevanje fizičnih IP naprav v simulator (bridge).

Vključena je tudi simulacija komunikacije med napravami (omrežni simulator) na osnovi UPnP (ob podpori UPnP knjižnice 'ClinkX'). Podpira protokole SOAP, SSDP, TCP/UDP, pa tudi OSPF in AODV. Simulacija radijske komunikacije upošteva vplive sten in ovir pri propagaciji signala.

Vsebinska pravila sistema se modelirajo z jezikom CADEL. Primer:

```
exLiving="Living Existence Sensor";
dvLight="Light";
LivingLamp1="LivingLamp1";
```

```
LivingLamp2="LivingLamp2";
LivFan="LivingFan";
clock="Example";
curtain="Curtain";
camera="SurveillanceCamera";
component="MusicPlayer";
fan="Fan";

ruleset(0) {
        if( exLiving.Alice ){
                dvLight.SetPower(1.0);
                LivingLamp1.SetPower(1.0);
                LivingLamp2.SetPower(1.0);
                LivFan.SetPower(1.0);
                clock.SetStatus(true);
                curtain.SetStatus(true);
                camera.SetPower(false);
                component.SetPower(true);
                fan.setPower(true);
        }
        if( !exLiving.Alice ){
                dvLight.SetPower(0.0);
                LivingLamp1.SetPower(0.0);
                LivingLamp2.SetPower(0.0);
                LivFan.SetPower(0.0);
                curtain.SetStatus(false);
                camera.setPower(true);
                component.SetPower(false);
                fan.setPower(false);
        }
}
```

Dokumentacijsko je orodje skromno podprto, ni vsebovanih novejših standardov (avtorji sicer navajajo, da bo podprt IEEE 802.11, Zigbee in Bluetooth.)

3.1.1.2 OMNeT++

Platforma: C++, Eclipse

Tip orodja: simulatorsko ogrodje

Namen: senzorska omrežja, brežična ad-hoc omrežja, internetni protokoli, P2P omrežja, optična omrežja, omrežni diski (SAN), performančno modeliranje

Tip licence: LGPL - Academic Public License (https://omnetpp.org/intro/license)

Zadnja verzija: 5.0 (15.4.2016)

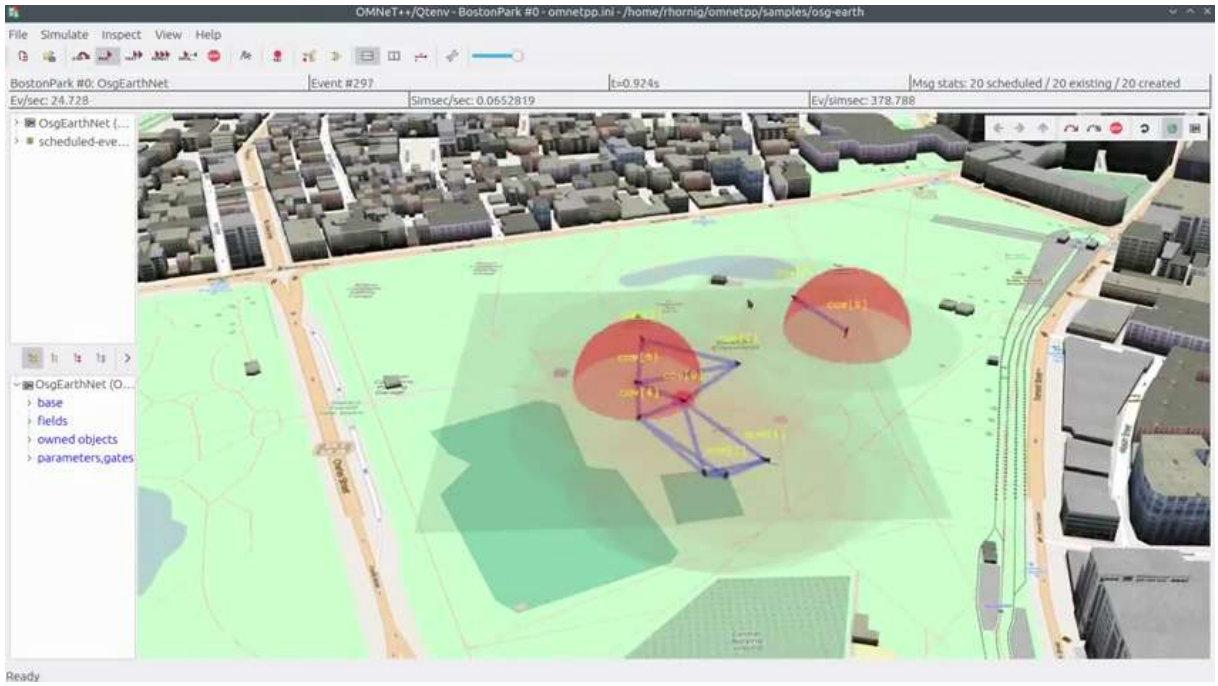
Spletna stran: https://omnetpp.org/

OMNeT++ je ogrodje za implementacijo diskretnih omrežnih simulacijskih modelov. V osnovi je bil to splošni simulacijski kernel, ki so mu dodali nekatere pomožne knjižnice (za generiranje naključnih števil - algoritem Mersenne Twister ipd.) za podporo implementacije

modelov, ki vključujejo vozlišča in povezave. Gre torej za simulator ožičenih in brezžičnih omrežij, ki pa je obenem tudi polno podprt splošni simulator.

Modeliranje v osnovi ni podprto z IDE, razen z uporabo ogrodja Eclipse. Izvajanje na vozliščih se definira ločeno z jezikom C++ (code behind). Konfiguracijo sistema oz. parametre simulacije nastavimo z ločeno ini datoteko, s čimer so ločeni parametri od samega modela. V primeru, ko eksperimentiramo na istem modelu z različnimi vhodnimi parametri, posledično ni potrebno spreminjati celotnega modela.

Rezultati so tekstovni in grafični, zapisujejo pa se tudi v log. V novi različici 5.0 so dodali modul za 3D vizualizacije z uporabo OpenGL, ki bodo sčasoma nadomestile stari modul za vizualizacije (Tkenv). [39] Z novo verzijo programa so bili vpeljani tudi materiali in konteksti (zemlja, teren, mesto), s katerimi lahko bolj realistično modeliramo pametna mesta, notranje prostore, pokrajino. Ta nivo je sicer podprt z API-jem OpenSceenGraph.



Slika 4: 3D izvedba modela v OMNeT++ (vir: https://www.youtube.com/watch?v=BjxVS9ExoI0)

Programska arhitektura je modularna, kar omogoča fleksibilne nadgradnje. Iz tega izhajajo številni raznoliki modeli ponujeni na spletni strani, kar je velika vrednost platforme OMNeT++. Modele potem glede na nišno načrtovano aplikacijo kombiniramo med seboj oz. jih razširjamo. Ti projekti nastajajo na ločenih vejah, neodvisno od osnovnega OMNeT++ projekta. Različni modeli in ogrodja, ki so jih razvile druge raziskovalne skupine za OMNeT++ platformo, se nahajajo na spletni strani OMNeT++.

Platforma uživa solidno podporo skupnosti. Na vsakoletnih delavnicah po različnih univerzitetnih mestih po Evropi predstavljajo nove protokole, modele, simulacijske postavitve

itd. [40] Zanimiv sklop modelov je bil predstavljen na OMNeT++ Summit 2015 konferenci (vir: https://www.youtube.com/watch?v=oO0l75BYFDI).

Dokumentacijska je dobro pokrita.

V naslednjih poglavjih bomo predstavili najbolj znane in za IoT zanimive razširitve.

3.1.1.2.1 INET

https://inet.omnetpp.org

Ogrodje INET je odprtokodna knjižnica za OMNET++. Vsebuje širok nabor protokolov in protokolnih skladov za namen omrežnega modeliranja.

Protokoli po kategorijah [41]

- Aplikacijski nivo: CBR/VBR, HTTP, FTP, DHCP, Video stream, Voice stream (VoIP),
- Transportni nivo: TCP, UDP, SCTP, RTP, RTCP
- Omrežni nivo: IPv4, IPv6, ICMPv4, ICMPv6, ARP, IGMPsv2, IGMPv3, MIPv6
- Routing: link-state routing, OSPF, OSPF, BGPv4, BGP, RIP, PIM
- Manet routing: AODV, DYMO, GPSR, DSDV, DSR, OLSR
- Multiprotocol label switching: MPLS, LDP, RSVP-TE
- Ožičene povezave: PPP, Ethernet, STP, RSTP,
- Brezžične povezave: 802.11, 802.1e, 802.15.4, 802.16e (WiMAX)
    - : različni mobilni modeli (TraCI)

INET se pogosto nadgrajuje (zadnja sprememba 17.6.2016)

Iz platforme INET potem izhajajo druge knjižnice in ogrodja, ki so razširjene za ožja področja modeliranja:

Ime ogrodja	Področje modeliranja	Zadnja spr.
SimulLTE	LTE omrežje (PDCP-RRC, RLC, multiplexing, RLCU PDUs buffering, AMC, scheduling policies, HARQ, realistični modeli kanala (interferenca, path-loss, fast fading, shadowing), X2 komunikacija, DAS.	nov. 2015
OS^3	satelitska komunikacija	avg. 2013
EbitSim	torrent	avg. 2012
OverSim	P2P	dec. 2012
ReaSE	omrežni vdori, hierarhične topologije	sep. 2011
NETA	omrežni vdori	jan. 2014
CAN model	CAN (controller area network) v cestnem prometu	apr. 2014
INETMANET	MANET	jun. 2014

Možne so poljubne kombinacije, npr. OverSim + ReaSE.

3.1.1.2.2 MiXiM

http://mixim.sourceforge.net

MiXiM je nabor knjižnic za modeliranje mobilnih in fiksnih brezžičnih omrežij (WSN, WBAN, ad hoc omrežja, omrežja vozil). Vključuje natančne modele za propagacijo radijskega signala, analizo interferenc, energetsko porabo in protokole za brezžično komunikacijo, mobilnost. Podpira protokole brezžičnih omrežij predvsem za fizični komunikacijski nivo povezav. Sama platforma praviloma ne nastopa kot samostojna, ampak gre dejansko za vmesni sloj - realistično knjižnico za nivo radijskih povezav, ki jo potem vključujejo v končne platforme. Knjižnica je npr. uporabljena v simulatorju Veins.

Zadnja verzija: 2.3 (marec 2013)

3.1.1.2.3 Veins

Platforma: Linux (priporočeno), Mac OS X, Windows
Razvijalec: različne raziskovalne skupine
Tip orodja: simulator povezanega prometa
Namen: komunikacije v cestnem prometu
Zadnja verzija: 4.4 (22.3.2016)
Spletna stran: http://veins.car2x.org/

Ogrodje za modeliranje komunikacije v cestnem prometu (inter-vehicle communication). Zgrajen je na osnovi SUMO (prometna simulacija) in MiXiM / OMNeT++ (omrežna simulacija). Simulatorja delujeta paralelno, povezana sta preko TCP povezave.

Podprti so modeli standardov IEEE 802.11p, IEEE 1609.4 DSRC/WAVE omrežni sloji, izbiranje wireless kanala in s tem povezan QoS, modeliranje interference (Two-ray model) in ovir (npr. vpliv stavb) pri propagaciji signala (radio shadowing). Ogrodje podpira tudi LTE omrežje (z uporabo SimulLTE).

Podpira uvoz scenarija OpenStreetMap, kar vključuje stavbe, hitrostne omejitve, število pasov, semaforjev in določena druga pravila.

Omogoča spreminjanje parametrov simulacije med izvajanjem v realnem času.

Na spletni strani so dostopne različne razširitve, npr. Veins LTE in npr. Plexe (avtomatsko zasledovanje vozil - platooing). [42]

Možne aplikacije so varnost, udobnost in učinkovitost vožnje.

3.1.1.2.4 Castalia

Ustanova: NICTA (National ICT Australia)
Tip orodja: simulatorsko ogrodje
Namen: WSN, WBAN, naprave z omejenimi viri
Zadnja verzija: 3.2 (30.3.2011)

Spletna stran: https://castalia.forge.nicta.com.au/

Castalia je razširitev ogrodja OMNeT++ za WSN in WBAN (IEEE 802.15.6) ter naprave z omejenimi resursi.

Avtorji so želeli izdelati ogrodje, ki bi čim bolj realistično posnemalo spodnje komunikacijske sloje brezžičnih omrežij, zato orodje ni omejeno na določeno strojno platformo.

Vključen je naprednejši stohastični modeli izgubnega kanala (path loss), implementiran na osnovi empiričnih podatkov. Dodatno obsega model radijske povezave - verjetnost sprejema podatkov (na osnovi SINR, velikosti paketov, tipa modulacije) in druge funkcije, ki so vezane na radijsko komunikacijo nizkoenergijskih naprav. Vključeni so določeni protokoli (MAC in routing). Na tej osnovi je možno evaluirati topologije, interferenco, porabo energije, obremenitve CPU. V model je možno vpeljati razne omejitve senzorjev (odstopanje meritev, šum).

Čeprav gre za starejši program, je precej uporabljen v drugih ogrodjih in pogosto citiran ter uporabljan tudi kot knjižnica v drugih simulatorjih (google skupina je videti precej aktivna).

3.1.1.2.5 Druge razširitve platforme OMNeT++

Ime ogrodja	Opis	Zadnja spr.
Numbat	Modeliranje WiMAX (IEEE 802.16) za IPv6	dec. 2011
NesCT	Izvajanje TinyOS aplikacij v okolju OMNeT++	

Na strani modelov in dodatkov za OMNeT++ (https://omnetpp.org/models/catalog?start=20) so med drugim navedeni tudi:

- ovire za model propagacije signala (https://omnetpp.org/models/catalog/download/33-models/2186-obstacle-extension-for-mf),
- protokoli za mobilni routing (FROMS, Directed Diffusion, Multicast Directed Diffusion, MSTEAM), (https://omnetpp.org/models/catalog/download/33-models/2201-routing-protocols-for-mobility-fw-2-02)
- modeli za optično omrežje (PhoenixSim, EPON),
- hibridna omrežja (optično/brezžično: INET-HNRL) in
- več drugih modulov in ogrodij.

3.1.1.3 CupCarbon [43]

Platforma: Java
Ustanova: LAB-STICC - University of Brest, Francija
Tip orodja: WSN simulator in orodje za prototipiranje
Namen: Pametna mesta
Tip licence: GNU-GPL
Zadnja verzija: 2.9.1 (april 2016)

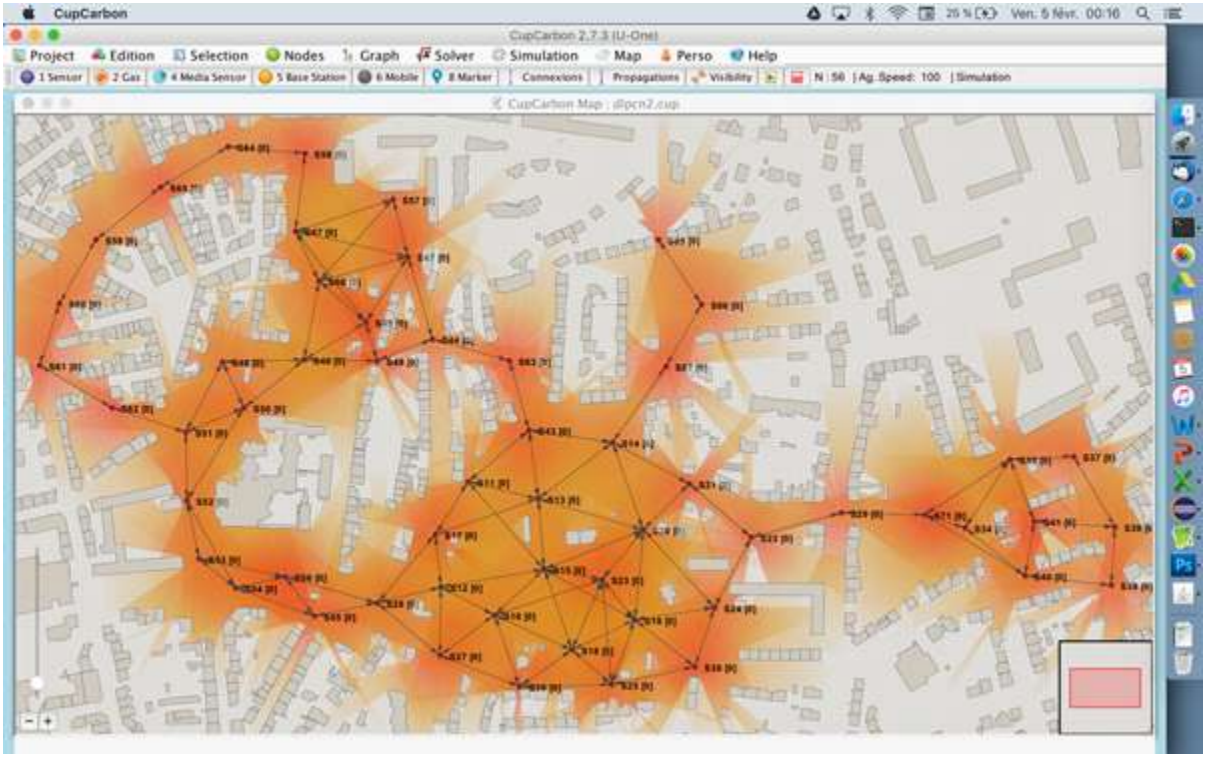
Spletna stran: http://cupcarbon.com/

Orodje CupCarbon je simulator WSN v kontekstu pametnih mest. Grafični vmesnik se v ozadju povezuje z OpenStreetMap API-jem in postavlja simulacijo v geolokacijsko okolje. Podatki OSM vključujejo podatke o stavbah, cestah. Posledično je s tem omogočeno realistično modeliranje za pametna mesta (npr. izris hiš in posledično ocenjevanje uspešnosti propagacije signala po prostoru).

V jedru sistema tečeta paralelno ločena simulatorja - simulator agentov (vozlišč) za izvajanje določene logike na napravah (npr. sprememba stanja in lokacije agenta ipd.) ter diskretni simulator (WiSeN) za izvajanje radijskega omrežja.

Vsako vozlišče, postavljeno na površino zemljevida, vsebuje točno informacijo o geolokaciji. Poleg tega pa še niz parametrov, ki jih je možno spreminjati med izvajanjem. Parametri so npr.: battery, rgauss, rotate (camera), readsensor, move (location). Torej vozlišča imajo v osnovi nabor lastnosti aktuatorjev, senzorjev, kar potem upravljamo s pomočjo skriptnega jezika Senscript. Jezik zajema minimalni nabor za koniguriranje in upravljanje simuliranih vozlišč v realnem času. Podprta je prevedba skripte v kodo za Arduino in pa ZigBee.

Izvedba na vozliščih je podprta na aplikacijskem nivoju s skriptnim jezikom. Podrobnejši protokolni skladi (razen IEEE 802.11 in 802.15.4) niso implementirani, razvoj protokolov tudi ni cilj tega orodja. Avtorji so nam pojasnili, da je cilj orodja pokrivati področja, ki jih drugi simulatorji ne pokrivajo. Obravnavajo predvsem učinkovitost komunikacije izgubnega radijskega kanala glede na različno število vozlišč. Zanima jih, kako se radijska komunikacija propagira po prostoru glede na njihovo umestitev med zgradbe. Kar se potem navezuje tudi na vprašanje vpliva radiacije na zdravje ljudi. S tem namenom so v orodje vključeni natančnejši modeli za interference (OFDM in Alpha stable distribution) in za propagacijo signala radijskega omrežja.



Slika 5: CupCarbon - vizualizacija propagacije signala za WSN omrežje, ki upošteva dejanske fizične ovire - zgradbe mesta (vir: https://www.youtube.com/user/EcoleUBO)

Simulator je uporaben za modeliranje večjega števila mobilnih vozlišč. Mobilnost vozlišč (agentov) je sicer podprta za vnaprej začrtano pot, vendar je v pomoč funkcionalnost, ki avtomatsko izrazi pot med dvema točkama na zemljevidu (izriše pot po cesti, pridobljeni iz OSM servisa).

V modelu je možno vpeljati kamero (motion senzor), katastrofične stimuluse kot požar, plin, destruktivni insekti. Podprti bodo tudi leteči objekti (npr. UAV). Izvedbo simulacije je možno tudi pohitriti in upočasniti. Vključen je tudi model energijske porabe naprav.

Projekt je obetaven, vendar mestoma deluje še kot prototip. Grafični vmesnik mogoče ni najbolj intuitiven (npr. za brisanje vozlišča je potreben backspace), posameznih vozlišč ni možno premikati. Dokumentacija je pomanjkljiva. Modeliranje je zgolj osnovno za posamezna vozlišča, ne masovno kot pri npr. simulatorjih konteksta. Ker je platforma arhitekturno razširljiva, bi bilo smiselno vpeljati kontekstni simulator prometa za podporo mobilnih agentov na vhod.

Moč tega orodja je v povezavi s projektom PERSEPTEUR, ki je bolj natančen simulator, namenjen za točnejšo simulacijo propagacije in interference v 3D urbanem okolju.

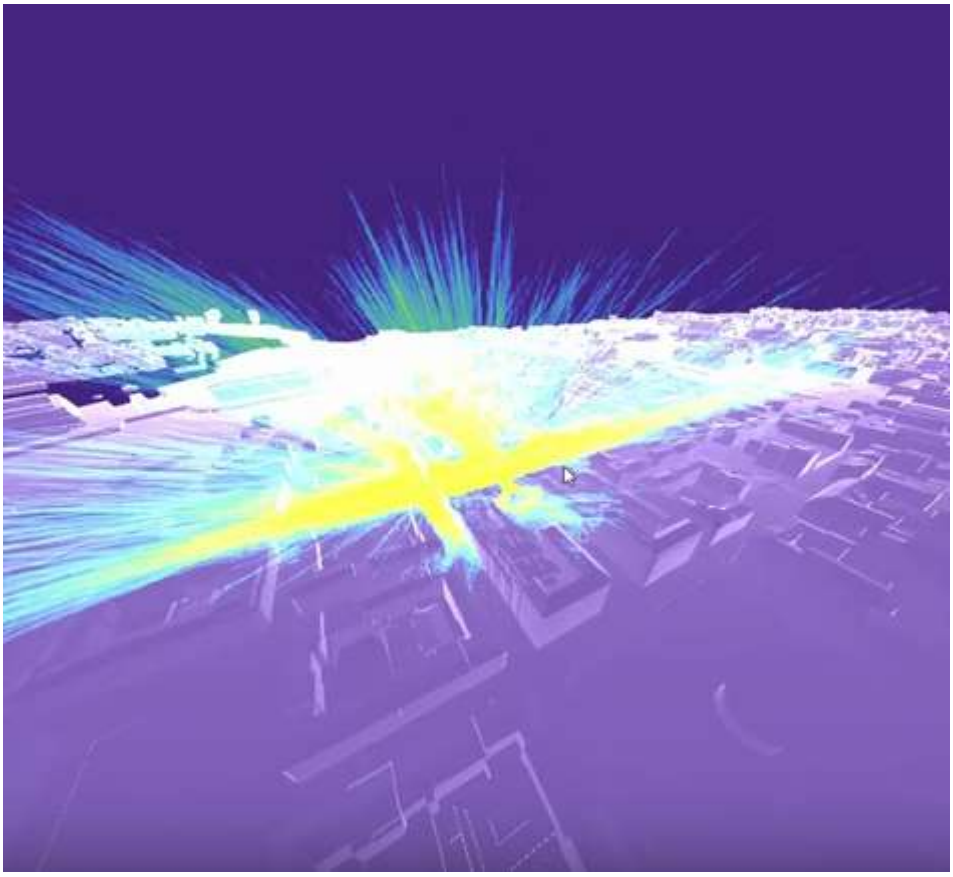
3.1.1.4 Persepteur [44] [45]

Ustanova: LAB-STICC - University of Brest, Virtualys, Francija
Tip orodja: WSN simulator
Namen: Analiza signala v 3D prostoru za pametna mesta

Zrelost programa: v razvoju

Standardni WSN simulatorji podpirajo 2D model, agenti in vozlišča postavljeni na ravnino, vizualizacija je omogočena za ptičjo perspektivo. Pri bolj podrobni analizi signala za neko WSN postavitev na določeni lokaciji v mestu je potrebno natančnejše orodje za simulacijo signala v prostoru.

Persepteur je simulator za modeliranje WSN in analizo radijskih signalov v 3D prostoru, razvit s strani avtorjev CupCarbon, za izvedbo simulacije uporablja skupno jedro. Gre za projekt v razvoju, videli smo le določene video prezentacije modelov ter izmenjali določene informacije z avtorji.



Slika 6: Persepteur - Vizualizacija propagacije signala v prostoru (vir: https://www.youtube.com/user/EcoleUBO)

Program izriše 3D vizualizacijo propagacije radijskega signala. Evaluira zanesljivost povezav glede na izhodno moč radijskega signala posameznih naprav. Omogoča vizuelno analizo propagacije signala po določenem območju, z upoštevanjem postavitve okoliških stavb, ulic ter drugih fizičnih ovir (drevesa, konstrukcije, hribi). Program izvaja kalkulacije interferenc glede na moč oddajnika in izbrani kanal v frekvenčnem prostoru. Vizualizacija zelo natančno prikaže odboje in zgostitve na določenih lokacijah. Simulator podpira tudi mobilne situacije po prostoru. Orodje podpira simulacije velikega števila vozlišč. Podobno kot pri CupCarbon, so stavbe izrisane na podlagi storitve OpenStreetMap.

Orodje prihrani komplikacije in stroške zaradi suboptimalne postavitve (vzdrževanje, energetska poraba). Smisel natančnejše analiza signala je med drugim tudi oceniti posledice določene postavitve na okolje, npr. z namenom zmanjšanja tveganja za zdravje ljudi. [44]

Za 3D jedro je uporabljeno grafično ogrodje Unity.

Avtorji so napovedali podporo tudi za protokolni sklad LoRa.

Propagacijo signala nazorno prikaže, zato je orodje uporabno tudi v študijskih programih. Analiza bi bila zanimiva tudi s stališča ustrezne postavitve agentov na določeni višini nad tlemi (npr. na določeno mesto na določeni zgradbi) za iskanje lokacije optimalnega pokritja. Orodje bi se bržkone izkazalo kot uporabno tudi za simuliranje aplikacij, ki vključujejo UAV.

3.1.1.5 NS-3

Platforma: Linux
Jezik: C++, Python
Tip orodja: omrežni simulator
Namen: protokoli, topologije, standardi..
Tip licence: GPLv2
Prva verzija: 2008 (začetek razvoja: 2006)
Zadnja verzija: ns-3.25 (marec 2016)
Spletna stran: https://www.nsnam.org/

NS-3 je diskretni simulator za komunikacijska omrežja. Vključuje modele za Wi-Fi, WiMAX, LTE sloje 1 in 2, različne druge protokole za IP-omrežja. Nove verzije simulatorja so objavljene približno na 3 mesece. [46]

Izvedba simulacije je podprta s skripto. Skriptni jezik je Python.

Vključuje vizualizacijo izvedbe simulacije z orodjem NetAnim.

Ns-3 se ponaša z modularno arhitekturo, zato so omogočene razširitve. Implementiranih je več dodatnih modelov in modulov, ki so na voljo na spletu. Ti gradniki so dobro dokumentirani in organizirani po kategorijah. [47]

Dodane so knjižnice za številne modele: antene, propagacijo signala, podprto je modeliranje stavbe in drugih ovir (različni materiali za absorpcijo signala), vključeni so modeli za porabo energije.

Med protokoli je podprt internetni protokolni sklad (IPv4, IPv6, TCP...), LR-WPAN protokoli (IEEE 802.15.4 - 6LowPAN), Wi-Fi, OpenFlow, LTE in WAVE in Wimax modeli, pa tudi model za podvodno brezžično komunikacijo UAN, protokoli za ad hoc mobilna omrežja (DSDV, DSR). Sklad Zigbee zaenkrat še ni posebej podprt.

Za vizualizacijo se uporablja modul NetAnim, omogočeno pa je tudi orodje za spektralno vizualizacijo.

Podprt je uvoz naključno generiranih topologij v večih formatih (Orbis, Inet, Rocketfuel).

NS-3 je možno povezati s fizičnimi napravami in drugimi simulacijami za in-the-loop izvajanje. Vgrajena je tudi podpora za izvajanje na večih procesorjih in porazdeljena izvedba simulacij preko P2P.

Model za mobilnost trenutno podpira samo kartezijski sistem koordinat, pri pretvorbi si je potrebno pomagati z ločenimi knjižnicami. 3D mobilnost (koordinata Z) ni podprta. Možna je povezava z zunanjimi orodji kot je BonnMotion, SUMO (npr. iTETRIS).

Sistem je dobro podprt z dokumentacijo.

NS-3 ni kompatibilen z njegovim predhodnikom NS-2, ampak gre za popolnoma na novo razvit simulator.

3.1.1.5.1 iTETRIS (Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions)

Platforma:Linux
Ustanova: iTETRIS Project Consortium
Tip orodja: simulatorska platforma
Namen: WSN za V2X (vehicle to vehicle, vehicle to infrastructure), zasnova V2X protokolov
Tip licence: GPLv3
Zadnja verzija: (program v času pregledovanja ni bil dostopen)
Spletna stran: http://www.ict-itetris.eu

iTetRIS je simulacijska platforma povezanega prometa V2X. Poudarek je na simulaciji ad hoc komunikacije med velikim številom vozil. Platforma je v osnovi kombinacija prometnega simulatorja SUMO in omrežnega simulatorja ns-3.

Podprti so komunikacijski standardi in modeli za WAVE (802.11p), WiMAX, DVB-H in UMTS. [48]

Možne aplikacije: dinamična izbira optimalne poti, prilagajanje hitrosti glede na situacijo, izvedba nujnih poti, uvedba avtobusnega pasu, zmanjšanje porabe goriva in znižanje emisij, obvladovanje gostote prometa, omejevanje hrupa, optimizacija pretočnosti prometa, omejevanje interference. [49]

Simulator je preko ustreznih vmesnikov možno povezovati z drugimi simulacijskimi orodji. IDE ni podprt, razen vizualizacije, ki jo omogoča SUMO. [50]

Razširitev projekta iTETRIS za področja učinkovitih algoritmov prometne signalizacije in nadzornih sistemov ter škodljivih emisij je projekt COLOMBO. [51]

3.1.1.5.2 NS-2

Platforma: C++

Tip orodja: omrežni simulator
Namen: protokoli, topologije, standardi..
Tip licence: GNU GPL 2
Zadnja verzija: 2.35 (november 2011)
Spletna stran: http://www.isi.edu/nsnam/ns/

NS-2 je predhodnik simulatorja NS-3 in ga omenjamo zaradi razširjenosti, vključen je v druge programske pakete na tem področju, za platformo je bila razvita obsežna množica modelov.

Slabost je kompliciranа uporaba, zasnova modela je zamudna, rezultati niso konsistentni.[52].

Na voljo so različna pomožna orodja za NS-2, med njimi naj omenimo npr. NRL Sensorsim - simulator senzorjev npr. za ogljikov monoksid, seizmični senzor, senzor zvoka. [53]

3.1.1.6 EASIM

Platforma: C++
Ustanova: Politecnico di Milano
Tip orodja: simulator
Namen: raba energije v pametnih prostorih (prodorni sistemi)
Tip licence: za akademsko uporabo
Zrelost programa: prototipna
Spletna stran: http://box.necst.it/easim.html

EASIM je simulator za področje učinkovite porabe električne energije za pametne stavbe. Aplikativne rešitve ponujajo dobre možnosti za izboljšanje učinkovitosti porabe energije, kar sovpada s področjem pametnega omrežja (smart grid). [54]

Simulira dinamiko električnih naprav (porabniki), proizvodnih virov (omrežje, solarni paneli), hrambe energije v akumulatorjih.

Simulator je zgrajen na osnovi platforme SystemC, ki je hiter diskretni simulator za elektronska vezja in podpira jezik C/C++ (simulirano kodo je zato možno izvajati na končnih napravah). Arhitekti sistema so želeli uporabiti kar to dobro delujoče orodje in ga prevesti na področje pametnih stavb. Izvedli so naslednjo iznajdljivo abstraktno prevedbo: vir energije v pametni stavbi je v izvornem simulatorju pomnilnik, baterija je medpomnilik (buffer), hišna naprava je agent, ki bere podatke (iz pomnilnikov). Na soroden način sta prevedena tudi modela komunikacije in energije. [55]

Opazujemo različne implikacije sistema (poraba, stroški) glede na različne konfiguracije oz. robne pogoje (postavljena pravila, izbira vira energije glede na čas v dnevu).

Na trgu sicer obstaja več HVAC simulatorjev, vendar so se pri simulatorju EASIM razvijalci bolj osredotočali na izvedbo hišnih naprav, kar se potem nadgrajuje v druge IoT simulatorje. [56]

EASIM je del večjega projekta BOX, ki se navezuje na različna področja pametnih stavb s ciljem večje učinkovitosti. [57] Druga področja znotraj omenjenega projekta so poleg simulatorja EASIM med drugim še: detekcija ljudi v prostoru (BlueSentinel), intuitivni uporabniški vmesniki (BildingRules), znižanje porabe energije za ogrevanje s pomočjo avtomatske izbire ustrezne barve svetlobe (SPELL), sistem za detekcijo nevarnosti in nesreč v pametnem domu (DANGER) in nenazadnje Pametni sistem za udobno bivanje (ThermoSense).

3.1.1.7 SimPy

Platforma: Python
Tip orodja: simulator
Namen: splošni
Tip licence: MIT
Zadnja verzija: 3.0.9 (junij 2016)
Spletna stran: https://simpy.readthedocs.org/

SimPy je splošni diskretni simulator zgrajen po inspiraciji simulatorjev Simula in Simscript. Scenarija (potek) simulacije je definiran v jeziku Python. Možno je modelirati aktivna vozlišča (ljudi, vozila, naprave) ter resurse (strežnike). Podprto je spreminjanje hitrosti izvajanja simulacije. Podprto je povezovanje z drugimi simulatorji in z drugimi resursi preko omrežnega vmesnika za izvajanje in-the-loop simulacij, vendar izključno v realnem času. [58]

Dokumentacija je solidno podprta. Python je za modeliranje zelo primeren. Ker gre za splošni simulator, je uporaben za simuliranje poljubnega aplikativnega nivoja ali domene IoT.

3.1.1.8 Ptolemy II

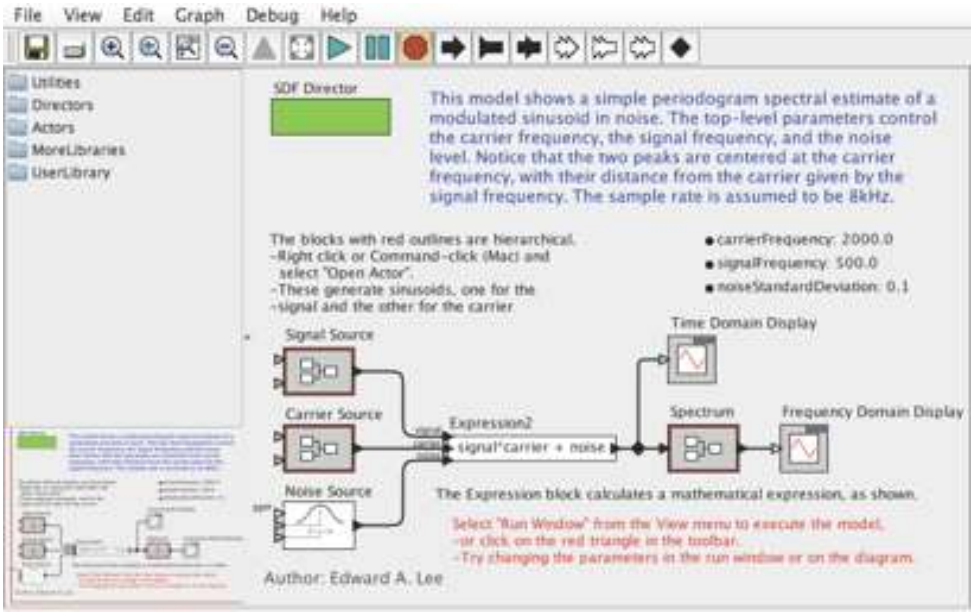
Platforma: Java
Ustanova: UC Berkeley
Tip orodja: simulator, orodje za prototipiranje, IDE
Namen: vgradni sistemi, sistemi realnega časa, procesiranje signalov, konkurenčnost, heterogeni sistemi
Tip licence: BSD
Zadnja verzija: produkcijska verzija 10.0.1 (december 2014), dnevne verzije dostopne v SVN
Spletna stran: http://ptolemy.eecs.berkeley.edu/

Ptolemy II je v osnovi splošni diskretni simulator, ki poleg osnovnih funkcij za diskretne simulacije vsebuje dodaten nabor gradnikov za simuliranje in prototipiranje algoritmov iz različnih področij elektrotehnike in računalništva.

Osnovni motiv razvoja, ki ga navajajo avtorji, je podpora simulacijam kompleksnih heterogenih sistemov. V modelu lahko kombiniramo raznolike stile modeliranja (končni avtomat, podatkovni tok, diskretni dogodki, skriptni jezik), tvorimo različne (heterogene)

modele podsistemov, ki jih potem med seboj povezujemo v bolj kompleksne heterogene modele. Primer je izvedba elektronskih enot (podsistemov) v avtomobilu, ki jih potem povežemo v skupno heterogeno simulacijo. [59]

Podprt je uporabniku prijazen grafični urejevalnik modela. Potek simulacije (scenarij) modeliramo tako, da med seboj logično povezujemo funkcijske elemente in jih parametriramo:



Slika 7: Ptolemy II (vir http://ptolemy.eecs.berkeley.edu/conferences/10/tutorial/index.htm)

Simulator je možno parametrirati tudi med izvajanjem. Arhitektura orodja je modularna in razširljiva. Razširitve so HyVisual, VisualSense, in Viptos. [60]

HyVisual podpira modele zveznega časa in hibridne modele (zvezne + diskretne). VisualSense vsebuje bogat nabor funkcij za modeliranje WSN, med drugim tudi porabo energije, senzorje, lokalizacijske strategije ipd. Simulator je uporaben izključno za WSN. Viptos (Visual Ptolemy and TinyOS) je povezava Ptolemy II in TOSSIM. Gre za prevajalnik grafičnih modelov v programske module nesC, katere je potem možno izvajati na TinyOS platformi in obratno - izvajati TinyOS aplikacije v hibridni simulaciji. [53]

Dokumentacija je dobro podprta. Čeprav produkcijska verzija že dalj časa ni bila zvišana, se v SVN sistemu tekoča razvojna verzija pogosto posodablja.

3.1.1.9 Shawn

Platforma: Java
Ustanova: Braunschweig University of Technology, University of Lübeck
Tip orodja: simulator
Namen: veliko število WSN vozlišč, porazdeljeni algoritmi
Tip licence: Thai open source

Zadnja sprememba: maj 2013
Spletna stran: https://github.com/itm/shawn/wiki/Shawn-Introduction;
https://github.com/itm/shawn https://sourceforge.net/projects/shawn/

Shawn je simulator za skalabilna WSN omrežja - velikega števila vozlišč (rang 100.000). Avtorji so prihranek dosegli tako, da so za komunikacijo in prenos vzpostavili približne modele namesto detajlno implementiranih protokolov (emuliranih), kot je to praksa pri drugih simulatorjih tega tipa. [61] Podoben pristop je sicer uporabljen tudi pri simulatorjih brezžičnih topologij. Vključen je tudi model senzorjev.

Orodje je uporabno za testiranje distribuiranih algoritmov in protokolov, evaluacijo učinkovitosti komunikacije in robustnosti WSN sistema velikega obsega.

Logiko na vozliščih implementiramo v jeziku C++. Simulator je enostavno parametrirati. Simulacija se proži iz komandne vrstice. Vizualizacija izvedbe je mogoča z razširitvijo.

3.1.1.10 SIDnet-SWANS [62] [63] [64]

Platforma: Java
Ustanova: Northwestern University
Tip orodja: simulator
Namen: WSN - komunikacijski in aplikacijski nivo
Tip licence: za akademsko in nekomercialno uporabo
Zadnja verzija: 1.5.6. (junij 2011)
Spletna stran: http://users.eecs.northwestern.edu/~ocg474/SIDnet.html

Simulator za WSN aplikacije. V osnovi je zgrajen iz enote za modeliranje protokolnega sklada (SWANS) ter enote, ki podpira modeliranje okolja in grafični vmesnik z animacijo izvedbe (SIDnet).

Omogočena je interakcija s simulatorjem med izvajanjem, spreminjanje hitrosti izvajanja. Vključuje model za porabo energije. Trenutna poraba je predstavljena intuitivno grafično v barvah neposredno na WSN območju. Implementirani so API-ji za povezovanje simulatorja z drugimi orodji in simulatorji. Arhitektura programa je modularna in odprta za razširitve. Dokumentacija je dobro podprta. Podpira veliko število vozlišč (rang milijon).

3.1.1.11 GrooveNet [64]

Platforma: C++ / Linux
Ustanova: University of Pennsylvania
Tip orodja: simulator
Namen: protokoli za V2X
Tip licence: GNU GPL
Zadnja sprememba: julij 2013
Spletna stran: https://github.com/mlab-upenn/GrooveNet

GrooveNet je simulator za izvedbo komunikacijskih ad hoc omrežij v prometu. Podpira večje število vozil (rang 1000). Na vsakem od vozlič je implementiran lasten model mobilnosti, izvedbe poti in komunikacije. Dodani so tudi modeli spreminjanja pasu, gostote prometa, modeli za GPS komunikacijo, sledenje vozil (platooing). Modularna zgradba omogoča razširitve modelov (model varnosti npr.). Nižji komunikacijski sloji so simulirani.

Vzpostaviti je mogoče hibridne modele, ki poleg simuliranih vozil vključujejo tudi fizična vozila. Komunikacija med fizičnim in simuliranim vozilom je sicer omogočena samo, če se obe vozili nahajata znotraj določenega območja vnaprej določene pokritosti. V simulaciji je možno spremljati tudi druge dogodke, ki jih proži sledilna naprava fizičnega vozila, npr. nujna sporočila in opozorila.

Vizualizacija simulacije je podprta z zemljevidom OpenStreetMap, ki ga je potrebno predhodno ročno uvoziti. Med izvajanjem so vozila predstavljena na mapi glede na trenutno GPS lokacijo vozila. Podprto je naključno generiranje postavitve vozil na modelirano območje. Pri izvedbi simulacije vozila upoštevajo omejitve hitrosti in semaforje (z uporabo modela semaforizacije).

Dokumentacija je solidno podprta.

3.1.1.12 Worldsens Simulator

Platforma: C
Ustanova: INRIA
Tip orodja: simulator, orodje za prototipiranje
Namen: WSN
Tip licence: CeCILL, GPLv2
Zadnja sprememba: januar 2012
Zrelost programa: eksperimentalna
Spletna stran: http://wsim.gforge.inria.fr

Worldsens Simulator je orodje za testiranje in prototipiranje WSN aplikacij. Sestavljen je iz simulatorjev WSim in WSNet.

WSim je simulator strojne platforme na nivoju instrukcij mikrokrmilnika. Podprta je emulacija za naprave s procesorjem TI MSP430 in nekatere druge. Celoten seznam podprtih strojnih naprav je objavljen na spletni strani. Na simulirani napravi je možno uporabiti različne operacijske sisteme - TinyOS, ContikiOS, FreeRTOS. [60].

WSNet je namenjen za simulacijo konteksta naprave in okolja. Vključuje dobro podprt nabor različnih modelov radijske propagacije, interference, modulacije, anten, modeli za mobilnost, porabo energije, fizikalni fenomen (ogenj), fizikalne količine (temperatura, vlaga). [53]

Na enem računalniku je možno simulirati rang 10-15 naprav, za simulacije višjega ranga pa je podprto izvajanje v računalniški gruči.

WSim in WSNet je možno uporabljati tudi ločeno, npr. za povezovanje v hibridnih platformah.

Grafični urejevalnik ni posebej podprt. Vizualizacija izvedbe je do neke mere podprta na nivoju WSim (LED indikatorji npr.).

3.1.1.13 Freedomotic

Platforma: Java
Razvijalec: Freedomotic
Tip orodja: simulator, orodje za prototipiranje, razvojno orodje
Namen: pametni prostori
Tip licence: GNU GPL 2
Zadnja verzija: 5.6.0-rc3 (1.7.2016)
Zrelost programa: beta.
Spletna stran: http://freedomotic.com/

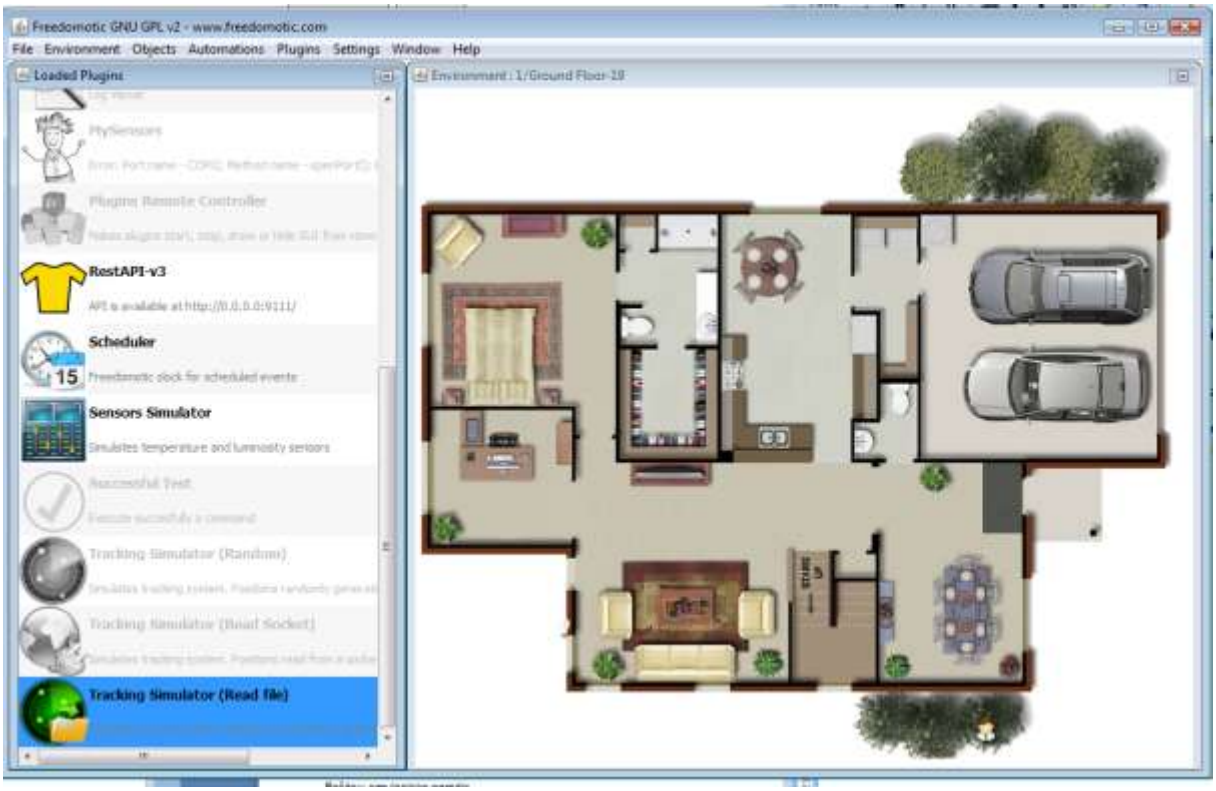
(Domotika = dom + informatika).

Freedomotic je razvojno okolje / simulator / kontrolnik za pametne prostore - domovi, pisarne, muzeji, šole, trgovine, fakulteta ipd.

Modeliranje se izvaja na najvišjem aplikacijskem nivoju. Orodje je enostavno in intuitivno za uporabo. Zasnujemo prostor na osnovi poljubne tlorisne slike. Z orodjem določimo stene in ovire (pohištvo). V model lahko postavimo tudi ljudi (mobilnost je omejena, ni direktne podpore za GPS pozicioniranje). Do določene mere je podprta mobilnost ljudi in sicer preko zunanjega vhoda ali zunanje datoteke, kjer je pot opisana.

V model je možno vključiti:

- različne končne IoT naprave, MQTT kliente, pametne luči,
- komunikacijske izvedbe Z-wave, BTicino OpenWebNet, Modbus RTU, Arduino mikrokrmilnik,
- IP kamera za zaznavo gibanja,
- zunanje storitve (Twitter) ali poljuben REST API,
- druge virtualne naprave (ki jih implementiramo kot vtičnike, npr. lahko druge naprave ali ročno izdelane kartice),
- dejanske fizične naprave preko funkcije autodiscovery.



Slika 8: Freedomotic

Gradniki za povezovanje so enostavno dostopni direktno iz programa (marketplace). Dodana je tudi podpora za direkten dostop do dodatnih gradnikov, ki jih ponuja online knjižnica. Razvijalci lahko tudi dodajajo svoje lastne naprave in storitve.

V model je možno dodajati poljubne storitve in ga povezovati z drugo programsko opremo ali fizično strojno opremo.

Podpira tudi logiranje sprememb, npr. v zunanjo bazo.

Slabosti: Mnogi gradniki šele bodo podprti. Omejeno simuliranje dejavnikov okolja (svetloba, temperatura). Ni dodano simuliranje fizikalnih količin in drugih dejavnikov okolja z uporabo porazdelitvenih funkcij.

3.1.1.14 Avrora [53]

Platforma: Java
Tip orodja: simulacijska platforma
Namen: razvoj vgradnih naprav, za izvedbo WSN za mikrokrmilnik AVR
Tip licence: open source
Zadnja verzija: 1.12 (avgust 2013)
Spletna stran: http://compilers.cs.ucla.edu/avrora/
http://avrora.cvs.sourceforge.net/avrora/
http://www.redcad.org/members/benhalima/azem/AZEM.html

Avrora je simulacijsko ogrodje za mikrokrmilnik AVR in WSN senzorske naprave MEMSIC Mica2 in MicaZ. Podprta je izvedba TinyOS aplikacij. Vključen je energetski model. Podpira večje število vozlišč (do 10.000).

Razširitev AvroraZ podpira tudi standardne protokole IEEE 802.15.4..

AZEM (AvroraZ + Energy + Mobility) je razširitev emulatorja AvroraZ, posebej za evaluacijo energetske porabe in mobilnosti WSN, omogoča tudi interaktivnost med izvajanjem simulacije.

Simulator je kot knjižnica vključen v simulator Cooja.

3.1.1.15 StreetLightSim

Ustanova: University of Southampton
Tip orodja: simulacijsko ogrodje
Namen: pametna javna razsvetljava
Zadnja sprememba: marec 2014
Zrelost programa: eksperimentalna
Spletna stran: http://www.streetlightsim.ecs.soton.ac.uk/

Javna razsvetljava je element, ki prispeva k izboljševanju prometne varnosti in omejevanju kriminalitete. Z različnimi ukrepi je možno znižati operativne stroške in zmanjšati vplive na okolje, npr. s prilagajanjem svetilnosti. Vpeljati je možno različne senzorje, s katerimi zaznavamo gibanje, karakteristike prometa (hitrost vozil, gostoto prometa), količino dnevne svetlobe, vremenske razmere. Drugi parametri delovanja so lahko obdobje dneva (ura, dan v tednu, praznik), geografska lokacija ipd. [65]

Raziskovalna skupina je implementirala simulator za ta namen. Uporabili so ga na konkretnem primeru za evaluacijo adaptivne aplikacije javne razsvetljave, postavljene na WSN omrežju. Našli so način, kako z različnimi ukrepi znižati porabo za 30% glede na že znane strategije zmanjševanja porabe energije ter za 90% glede na konvencionalne sheme.

Simulator StreetLightSim je zgrajen na osnovi OMNeT++ (MiXiM) in SUMO (deloma tudi Veins). Dodani so modeli za porabo energije ter modeli za porazdelitev porajanja prometa glede na dnevna in letna povprečja (zvečer, ponoči oz. poleti, pozimi).

3.1.1.16 DEUS

Platforma: Java
Ustanova: University of Parma
Tip orodja: simulator
Namen: splošno, skalabilne situacije
Tip licence: GNU GPL v2
Zadnja verzija: 0.6.0 (junij 2014)

Zrelost programa: začetna
Spletna stran: http://dsg.ce.unipr.it/?q=node/29 https://github.com/dsg-unipr/deus/ https://code.google.com/p/deus/
Navodila za namestitev: https://github.com/dsg-unipr/deus/wiki/DEUS-Tutorial

DEUS je splošni simulator za kompleksne sisteme. Na osnovi DEUS je zgrajen OSMobility. Ključne podatkovne strukture simulatorja so: vozlišča (agenti/entitete - ljudje, živali, roboti, naprave), dogodki (porajanje/ponikanje vozlišč, interakcije med vozlišči, interakcije vozlišč z okoljem, dnevniki) ter procesi (stohastični in deterministični proces).

Model je zapisan v XML. Vključen je tudi grafični urejevalnik. [29]

3.1.1.17 OSMobility

Platforma: Java
Ustanova: University of Parma
Tip orodja: simulator
Namen: mobilnost (avtomobili, pešci, kolesarji,...)
Zadnja verzija: 1.0 (2014?)
Spletna stran: http://dsg.ce.unipr.it/?q=node/88
https://code.google.com/archive/p/osmobility/

OSMobility omogoča simulacijo prometa v kontekstu mesta. Vključuje različne agente v urbanem okolju kot so avtomobili, pešci, kolesarji. Za vizualizacijo simulirane situacije uporablja prosto dostopno geolokacijsko storitev OpenStreetMap, ki vsebuje točne infrastrukturne definicije - pozicije hiš, trase cest, kolesarskih stez, pločnikov. Posledično OSM platforma podpira realistično modeliranje poti pešcev, vozil in koles znotraj njim namenjenih območij (pešci hodijo po pločniku, kolesarji se vozijo po kolesarskih stezah, avtomobili po cestah in avtocestah). Podprt je SIL.

Podpira model mobilnosti Fluid Traffic Model (FTM) ter izgradnjo specialnih modelov za specifične situacije. Omogočena je tudi podpora za dinamično iskanje optimalnih poti. [28]

Trenutno stanje projekta je neznano. Čeprav naj bi bila koda odprta, ni dostopna na objavljeni spletni lokaciji. Poskušali smo stopiti v kontakt z avtorji, žal odgovora nismo prejeli.

3.1.1.18 MAMMotH

Ustanova: Data Communications Sotware Lab, Aalto University, Finland
Tip orodja: omrežni simulator
Namen: masovno število vozlišč
Spletna stran: http://mammoth.fi/

MAMMotH je projekt vzpostavitve simulatorja za masovno število vozlišč (več milijonov hkratno delujočih vozlišč), s časovno kompleksnostjo O(n). [66] Projektna skupina si je za cilj zastavila simulacijo 20 milijonov vozlišč na gruči računalnikov.

Na spletu žal ni izvorne kode, projekt tudi ni bil osvežen od leta 2013, o tem simulatorju ni bilo moč najti nobenih novejših člankov. Avtorji so nam sporočili, da je projekt zaključen.

3.1.1.19 DEVSimPy

Platforma: Python
Ustanova: SPE, University of Corsica Pasquale Paoli
Tip orodja: simulator
Namen: splošni, izvedba DEVS modelov
Tip licence: GPL v.3
Zadnja verzija: 2.9 (2016)
Zrelost programa: eksperimentalna
Spletna stran: https://github.com/capocchi/DEVSimPy

DEVS (Discrete Event System Specification) je formalizem za opisovanje modelov sistemov diskretnih dogodkov. Uporablja se na področju simulacij v IKT [67], pa tudi na drugih področjih, kjer imamo opravka z bolj kompleksnimi sistemi (hidravlično omrežje, gozdni požari, področja genetike).

DEVSimPy je odprtokodno ogrodje in GUI za modeliranje skladno s formalizmom DEVS, prevedbo v jezik Python ter izvedbo simulacije takšnih modelov. Jedro tega orodja je odprtokodni API PyDEVS. [68]V sam programski paket so dodani različni modeli in orodja. Omogoča tudi spreminjanje parametrov med izvajanjem simulacije.

Sehili et al. [69] so predstavili tudi metodologijo razvoja IoT aplikacij z uporabo DEVSimPy in middleware sistema WComp. Modele razvite v DEVSimPy je možno uporabiti na platformi WComp brez prevajanja kode.

Simulator je možno uporabiti za različne segmente kompleksnih aplikacij kot so IoT, pa tudi izvedbe vgradnih naprav in porazdeljene situacije. Orodje je zanimivo ravno zaradi podpore formalizmu DEVS, kar omogoča prevedljivost modelov in integrabilnost z drugimi orodji podprtimi z DEVS formalizmom.

Navodila za uporabo so v francoskem jeziku. Na youtube je objavljenih nekaj prezentacij programskega orodja.

3.1.1.20 ZBOSS Network Simulator

Platforma: Linux
Razvijalec: DSR Corporation
Tip orodja: omrežni simulator

Namen: Protokolni sklad ZigBee
Tip licence: GPL
Zadnja verzija: 1.0
Spletna stran: http://zboss.dsr-wireless.com

ZBOSS je omrežni simulator za ZigBee protokolni sklad. Uporaben je za razvoj aplikacij na ZigBee platformi brez potrebe po fizičnih napravah. Podpira platforme MCU 8051, ARM 11, Linux x86 ter radijske module Texas Instruments TI CC253x ter UBEC 2400 in 2410. Za dolpoteg se je potrebno predhodno registrirati.

Na voljo je tudi verzija ZBOSS 2.0 s podporo novejšim strojnim platformam in ZigBee protokolom, vendar licenca ni odprtokodna.

3.1.1.21 Automatski

Platforma: Java
Razvijalec: Automatski
Tip orodja: simulacijsko ogrodje, knjižnica modelov
Namen: IoT
Tip licence: GPL + komercialna
Zadnja sprememba: avgust 2015
Spletna stran: https://github.com/adityayadav76/internet_of_things_simulator

V osnovi je to simulacijsko ogrodje, ki pa vključuje modele različnih protokolov, ki se uporabljajo v IoT na višjih aplikacijskih slojih: AMQP, CoAP, LWM2M, MQTT, Rest, SmartM2M, UDP, Websocket, XMPP.

Namenjen naj bi bil za simulacijo obsežnega števila naprav (rang milijon naprav).

Vrednost tega produkta bi bila predvsem v knjižnici protokolov.

3.1.1.22 Desmo-J

Platforma: Java
Ustanova: University of Hamburg
Tip orodja: simulacijsko ogrodje
Namen: splošni
Tip licence: Apache 2.0
Zadnja verzija: 2.5.1c (november 2015)
Spletna stran: http://desmoj.sourceforge.net/home.html

Splošno simulacijsko ogrodje, ki je zasnovano na objektnem principu. Vključeni so osnovni razredi za stohastično modeliranje, na tem pa abstraktne razrede za vzpostavitev modela za konkretno domeno. Dokumentacija je dobro podprta.

3.1.1.23 ArduPilot

Platforma: Windows, Linux
Tip orodja: programska platforma
Namen: avtomatsko upravljanje UAV, vozil in plovil
Tip licence: GPLv3
Zadnja verzija: avgust 2016 (različne verzije različnih orodij)
Spletna stran: http://ardupilot.org/ardupilot/index.html

ArduPilot/APM je odprtokodna programska oprema za PC podporo avtomatskega upravljanja radijsko vodenih letalnikov, multikopterjev, helikopterjev, vozil in plovil. S priloženim planerjem definiramo pot (misijo) in jo instaliramo na kontrolnik vozila za končno izvedbo na terenu.

Sistem je bil v osnovi zgrajen na mikrokrmilniku Arduino, sčasoma pa se je podpora razširila na specializirane vgradne plošče za avtomatsko upravljanje vozil kot so Pixhawk, Arsov AUAV-X2, Erle-Brain, NAVIO+. Okoli sistema ArduPilot pa se je pojavila druga kompatibilna strojna oprema: Minim OSD, 3DR telemetry idr.

V programski opremi je vključen tudi simulator, ki deluje na dva načina. Način HIL povezuje program začrtane poti APM planer ter strojno opremo kot so RC naprava in kontrolnik.

Pri čistem SIL načinu izvedbe simulacije je simuliran tudi kontrolnik. Ta pristop je enostavnejši za konfiguracijo, v celoti se izvaja na PC, omogoča razhroščevanje itd. S pomočjo python skript je možno tudi avtomatsko testiranje.

V simulacijo lahko vključimo tudi zunanji simulator letenja. Simulatorji letenja vključujejo modele okolja - turbulence npr. - na takšen način je možno varno in realistično testirati najrazličnejše scenarije. Za simulacijo letenja se priporočajo odprtokodni JSBSim ali FlightGear, pa tudi last_letter, kot licenčni pa X-Plane. Za kopterje se uporablja orodja MAVProxy ali ROS/Gazebo. Omenjen je tudi CRRCSim, ki vključuje vrsto modelov kril in podporo za kopterje. [103]

3.1.1.24 Mosaik

Platforma: Python
Razvijalec: OFFIS, Nemčija
Tip orodja: simulacijska platforma, kosimulator
Namen: pametno omrežje (smart grid)
Tip licence: GNU-LGPL
Zadnja verzija: 2.2.0 (16.2.2016)
Spletna stran: https://mosaik.offis.de/

Pametno omrežje sestavljajo različni viri energije, ki pogosto niso zanesljivi (sonce, veter in druge metode žetve energije). V omrežju imamo različne porabnike (aparature, naprave), kjer

pa veljajo določene zakonitosti kot je pričakovana poraba v določenem dnevu ob določenem času. Temu primerno vsakodnevna cena na borzah občutno niha. S povezovanjem kontrolnikov električnih avtomobilov, pametnih števcev, pametnih virov energije v kompleksne IoT aplikacije je možno bolje prognozirati porabo in proizvodnjo (iz nezanesljivih virov), temu ustrezno prilagajati shranjen potencial v baterijah ali akumulacijskih jezerih ipd. [70] Na področju električnega omrežja obstaja vrsta možnosti, kako uporabiti komunikacijsko tehnologijo, senzoriko in aktuatorje na pametni mreži, da bi vzpostavili bolj ekonomično izrabo energije, bolj uravnoteženo porazdelitev po prostih kapacitetah in boljši izkoristek.

Orodje Mosaik je kosimulator za povezovanje obstoječih 3rd-party simulatorjev in simulacijskih orodij za pametna omrežja. Funkcija Mosaika je koordinacija izvedbe porazdeljene simulacijske postavitve v kompleksne realistične simulacijske scenarije. V model povežemo različne ločene domenske simulatorje za pametna omrežja: fotovoltaika, elektrarne, prenos električne energije, gospodinjske in industrijske odjemalce. Ob tem pa še druga orodja za analitiko (npr. obremenitve omrežja), logiranje ipd. Povezovanje je omogočeno preko API-jev, format podatkov je JSON. Mosaik izvaja funkcijo posrednika podatkov med ločenimi simulacijskimi procesi, izvaja periodične poizvedbe na določenem procesu, zapisuje vmesne rezultate na analitično enoto, sinhronizira izvedbo ločenih paralelno izvajanih simulacij na osnovi scenarija, ki ga koordinira.

Orodje je zgrajeno na osnovi simulatorja SimPy. V osnovi gre torej za samostojno simulacijsko jedro na Pythonu, ki omogoča tako izvedbo scenarija (koordinacijo zunanjih procesov) kot tudi izvedbo morebitne notranje ločene simulacijske logike. Možno je tudi uvoziti druge Python simulatorje in izvajati sinhrono simulacijo (brez potrebe po programiranju API-jev in omrežne režije) [71]

Vir [72]omenja tudi različne simulatorje električnega omrežja: OpenDSS, GridLAB-D, PowerFlow, ki jih iz našega stališča potem smatramo kot simulatorje konteksta (električnega omrežja) in vizualizacije.

Podoben kosimulator je tudi MESCOS. [73]

3.1.1.24.1 Mosaik + OMNeT++

Mosaik v osnovi predpostavlja idealen komunikacijski kanal. Za simulacijo realne postavitve podprte z WSN je raziskovalna skupina iz Univerze v Bremnu preizkušala način za povezavo simulatorja WSN, torej OMNeT++ in Mosaik. [72]

Raziskovalci so naleteli na praktične izzive na račun dejstva, da je Mosaik simulator diskretnega časa, OMNeT++ pa simulator diskretnih dogodkov in ni namenjen zunanjemu krmiljenju. Sinhronizirano delovanje je možno doseči z dograditvijo na nivoju jedra

OMNeT++ in drugimi načini. Predvsem je pomembno, da izvedba poteka na aplikacijskem nivoju komunikacije, nižje nivoje pa prepustimo v izvajanje OMNeT++.

3.1.1.25 TRMSim-WSN (Trust and Reputation Models Simulator for Wireless Sensor Networks)

Platforma: Java

Ustanova: Universidad de Murcia

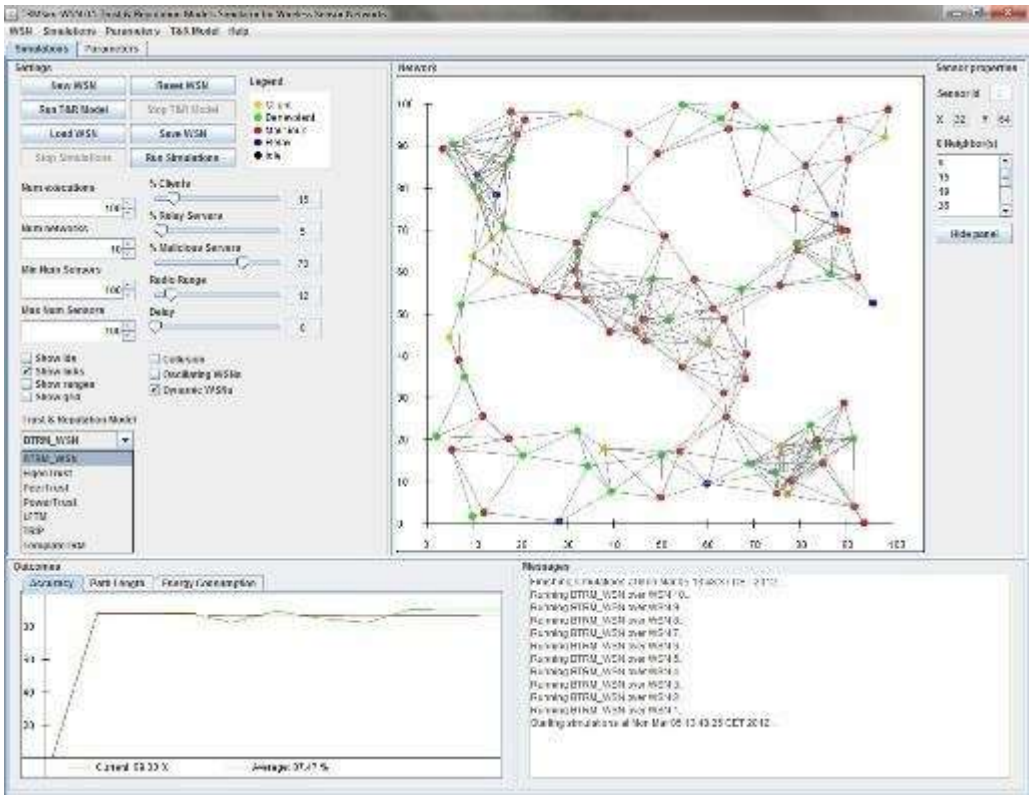
Tip orodja: simulator

Namen: algoritmi zaupanja in ugleda za WSN omrežja

Tip licence: GNU GPL 2.0

Zadnja verzija: 0.5 (marec 2012)

Spletna stran: http://ants.inf.um.es/~felixgm/research/trmsim-wsn/index.php



Slika 9: TRMSim-WSN (vir: https://sourceforge.net/projects/trmsim-wsn/)

Simulator za evaluacijo algoritmov zaupanja in ugleda v WSN omrežjih. Določeni algoritmi so že priloženi. Podprt je grafični vmesnik in vizualizacija poteka simulacije. orodje je odprto za nadaljnje razširitve.

3.1.1.26 Sensor Security Simulator (S3)

Platforma: Visual Studio (C++)

Ustanova: Faculty of Informatics, Masaryk University

Tip orodja: simulator

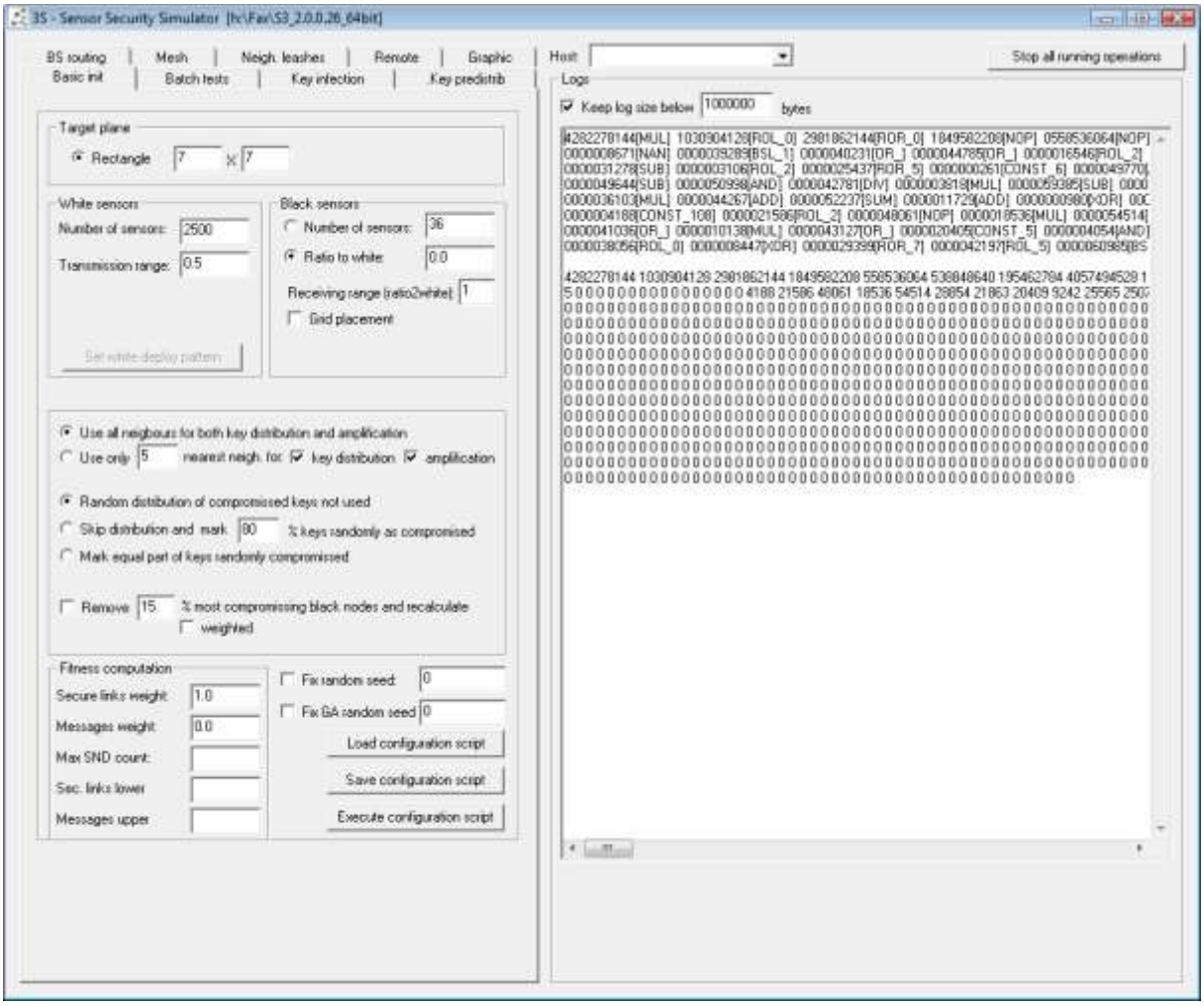
Namen: varnost WSN omrežij

Tip licence: prosto dostopna koda

Zadnja verzija: 2.0.0.26 (september 2012)

Spletna stran: http://www.fi.muni.cz/~xsvenda/s3.html

Sensor Security Simulator je namenjen evaluaciji najrazličnejših varnostnih vidikov WSN omrežja. Dodani so različni specialni varnostni algoritmi in protokoli. Simulator je optimiziran za simulacije velikih omrežij, t.j. rang več tisoč vozlišč. Takšna skalabilnost je dosežena na način, da so podrobnejši standardni komunikacijski protokoli rešeni algoritemsko (z modelom). Vključen je bogat grafični vmesnik, omogočena je konfiguracija s skripto. Podprta je tudi povezava s simulatorjem Matlab.



Slika 10: Sensor Security Simulator (S3)

3.1.1.27 WSN Localization Simulator

Platforma: .NET

Tip orodja: simulator

Namen: podpora razvoju lokacijskih algoritmov za WSN

Tip licence: CPOL

Zadnja verzija: 2.1 (junij 2013)

Spletna stran: http://www.codeproject.com/Articles/606364/Wireless-Sensor-Network-Localization-Simulator-v

WSN vozlišča so varčne naprave, zato je uporaba GPS za lokalizacijo preveč potratna možnost, poleg tega pa omejena samo na zunanje okolje. Za namen pozicioniranja varčnih senzorskih naprav se uporablja posebne algoritme, ki pa so samo relativno natančni (npr. v izračun so vključena tudi druga vozlišča). To simulacijsko orodje vzpostavlja WSN omrežje, vključuje modele energije, propagacije signala in mobilnosti ter se uporablja za evaluacijo takšnih algoritmov. Privzeto je v programski paket dodanih več takšnih znanih algoritmov. Simulirati je možno večje število vozlišč. Orodje je enostavno za uporabo. Programska arhitektura simulatorja je dobro dokumentirana, zato nudi dobro osnovo za razširitve ali izgradnjo novega simulacijskega orodja.

3.1.2 Simulatorji konteksta (modeliranje agentov)

S simulatorji te vrste modeliramo dinamiko v sistem. Vpeljemo množice uporabnikov in drugih subjektov (modeliranje agentov), pridobimo pa tudi grafično predstavitev situacij (modeliranje konteksta).

3.1.2.1 Netlogo [9]

Platforma: Java
Tip orodja: simulator agentov
Namen: naravni in socialni fenomeni
Tip licence: GNU GPL
Zadnja verzija: Verzija za MacOS, Windows ali Linux: 5.3.1. (februar 2016) ali Spletna verzija
Spletna stran: http://ccl.northwestern.edu/netlogo/index.shtml

Netlogo je orodje za modeliranje in simuliranje situacij, v katerih nastopajo različni agenti - ljudje, roboti, avtomobili, živali, rastline ipd. in zunanji dejavniki (t.i. Agent Based Model).

Lastnosti oz. vedenja agentov so definirane proceduralno s pomočjo programskega jezika, ki je zasnovan po vzoru jezika Logo (od tod ime Netlogo). [74] Med naborom rutin so med drugim tudi porazdelitvene funkcije (eksponentna, Poissonova, gamma, normalna itd.), funkcije vhoda in izhoda (zapis v zunanje datoteke), funkcije za modeliranje topologij ipd. Vključeno je posebno grafično orodje za oblikovanje dinamike sistema (modeliranje delovnega toka).

Model je možno podpreti tudi z grafičnimi GUI kontrolami. Na risalno podlago dodajamo grafične kontrole kot so drsniki, gumbi, izhod, kar povežemo na parametre simulacije in skriptno izvedbo. Ta funkcionalnost spominja na prototipno orodje. Med samo izvedbo simulacije interaktivno spreminjanje parametrov žal ni podprto, je pa možno spreminjati hitrost izvajanja. Vizualizacija je podprta tudi v 3D.

Orodje je na vtis preprosto in intuitivno za uporabo. Dokumentacija je dobro podprta. Nadvse praktična je bogata knjižnica že pripravljenih modelov, ki so urejeni po najrazličnejših raziskovalnih področjih (matematika, računalništvo, biologija, kemija, igre, psihologija, družbene vede, filozofija itd.). Poleg tega je na voljo tudi spletna zbirka modelov uporabnikov (Modelling commons). V zbirki najdemo primere za WSN in pametni prostor.

Orodje bi bilo za IoT uporabno tako za simuliranje primerov uporabe na aplikacijskem nivoju kot tudi simuliranje določenih protokolov in topologij, pa tudi kot generator podatkov, ki jih uporabimo na vhodu drugih simulatorjev.

Možne so tudi različne razširitve za povezave orodja navzven (ActiveMQ, BDI, FIPA, MySQL, MATLAB, Arduino itd.).

Spletna stran: https://github.com/NetLogo/NetLogo/wiki/Extensions.

3.1.2.1.1 Druga orodja na področju modeliranja agentov

Druga orodja iz področja modeliranja agentov bi bila bržkone tudi uporabna za simulacije agentov za IoT, vendar bi pregledovanje teh orodij presegalo okvire tega dela. Zainteresirani bralec lahko preveri platforme, ki se največkrat omenjajo npr. Repast Simphony in posebej Jade. Npr. Slednji ponuja uporaben repozitorij programskih razširitev simulatorja: http://jade.tilab.com/download/add-ons/

3.1.2.2 Siafu

Platforma: Java
Tip orodja: simulator konteksta
Namen: generiranje prostora in agentov
Tip licence: GPL
Spletna stran: http://siafusimulator.org
Testi: stestirano in delujoče
Zadnja sprememba: 11.12.2014

Siafu je simulator gibanja agentov - ljudi, vozil, živali. Vsak agent ima v vsakem trenutku informacijo o poziciji v prostoru in tudi druge poljubne atribute. Gre torej za generator pozicijskih in drugih podatkov za uporabo v simulacijskih modelih IoT ali npr. za strojno učenje. Izhod tega orodja se lahko poveže na druge simulatorje.

Ponuja tudi vizualizacijo v živo 2D iz ptičje perspektive, torej je možno tudi nivo same vizualizacije uporabiti v navezavi z drugimi simulatorji za IoT (npr. Cooja ali OMNeT++).

Orodje poskuša kar najbolje posnemati agente - njihovo porajanje in gibanje. Podpira nočni in dnevni čas. Podprta je pohitritev oz. upočasnitev časa izvajanja simulacije.

Nad grafično shemo modeliranega prostora je možno definirati omejitve - poti ljudi, po katerih se gibljejo, stavbe, v katere vstopajo, ceste, po katerih vozijo avtomobili. Pri modeliranju notranjih prostorov so omejitve gibanja stene, oprema, pohištvo itd.

Slika 11: Siafu

Priloženi so tudi vnaprej definirani modeli: pisarna in nekaj modelov mestnih središč, ki jih je mogoče dodatno prilagoditi.

Orodje ni namenjeno za modeliranje naprav, servisov in aplikacij (API, PaaS ipd.). Dodana vrednost simulatorja je v povezovanju z drugimi simulatorji. Primer integracije tega okolja v širši simulacijski kontekst je npr. DiaSim oz. DiaSuite [11] (op. orodji nista prosto dostopni).

3.1.2.3 SUMO (Simulation of Urban MObility)

Platforma:C++
Ustanova: German Aerospace Center, Institute of Transportation Systems
Tip orodja: simulator konteksta
Namen: generiranje cestnega prometa
Tip licence: GNU GPL
Zadnja verzija: 0.27.1 (12.7.2016)
Spletna stran: http://sumo.dlr.de/

SUMO je simulator prometa za kompleksna cestna omrežja. Omogoča natančno modeliranje posamičnih vozil, pešcev in sredstev javnega transporta. V model vključimo pravila na

določeni trasi (smer vožnje, dolžina intervalov na semaforju, porazdelitve porajajočega prometa). Preko razširitve ACTIVITYGEN je možno generirati lastnosti kot so čas obratovanja služb, šol, opredelitev prostega časa ter komutacijo - hoja, kolo, avtomobili in avtobus. [75]

Podpira tudi določene emisijske modele. V izvedbo je mogoče vključiti različne algoritme za iskanje poti, ki se dinamično prilagajajo npr. v primeru zastojev ipd. Vizualizacija poteka simulacije je omogočena iz ptičje perspektive. Omogočen je uvoz formatov OpenStreetMap, VISUM, VISSIM, NavTeq. Podprto je upravljanje simulacije na daljavo v realnem času s pomočjo nabora API-jev. [76]

Takšno orodje v osnovi služi planerjem prometne infrastrukture ali npr. z namenom zmanjševanja onesnaževanja, vendar se navezuje tudi na računalništvo v primeru pametnega mesta ali npr. pri V2X aplikacijah (VANET). Za te postavitve potrebujemo bolj precizen generator konteksta - simulator prometa.

Različna ogrodja, ki povezujejo SUMO z omrežnimi simulatorji, tvorijo on-line loop preko TCP povezave. [28] SUMO v teh primerih kot strežnik in se ga upravlja od zunaj v realnem času. Takšna orodja so iTETRIS, Veins in TraNS.

V povezavi z OpenStreetMap je potrebno še omeniti eWorld, ki podpira uvoz OSM map v SUMO simulator in prepoznavo objektov na mapi (semaforji). Področja na mapi je možno dodatno obogatiti z lastnostmi kot npr. določiti pogostost dežja, megle, snega. Dostopno na http://eworld.sourceforge.net.

3.1.2.4 BonnMotion

Platforma:Java
Tip orodja: generator mobilne poti
Namen: mobilni podatki za uporabo v drugih simulatorjih
Tip licence: GNU GPL
Zadnja verzija: 3.0.1 (april 2016)
Spletna stran: http://sys.cs.uos.de/bonnmotion/

BonnMotion je orodje za generiranje mobilnega scenarija oz. poti na osnovi večih podprtih mobilnih modelov (Random Waypoint, Random Walk, Gauss Markov, Manhattan itd.). Scenarije je možno izvoziti in potem uporabiti v simulatorjih kot so ns, GloMoSim, COOJA, MiXiM, ONE.

3.1.3 Virtualizatorji

Virtualizator vzpostavi navidezno IoT napravo v omrežju. Takšna instanca vsebuje realističen nabor vmesnikov in je videti kot prava fizična naprava.

3.1.3.1 DPWSim

Platforma: Java

Ustanova: Telecom SudParis

Tip orodja: prototipno in testno orodje, simulator

Namen: virtualne DPWS naprave

Tip licence: ? (koda je dostopna v izvorni obliki)

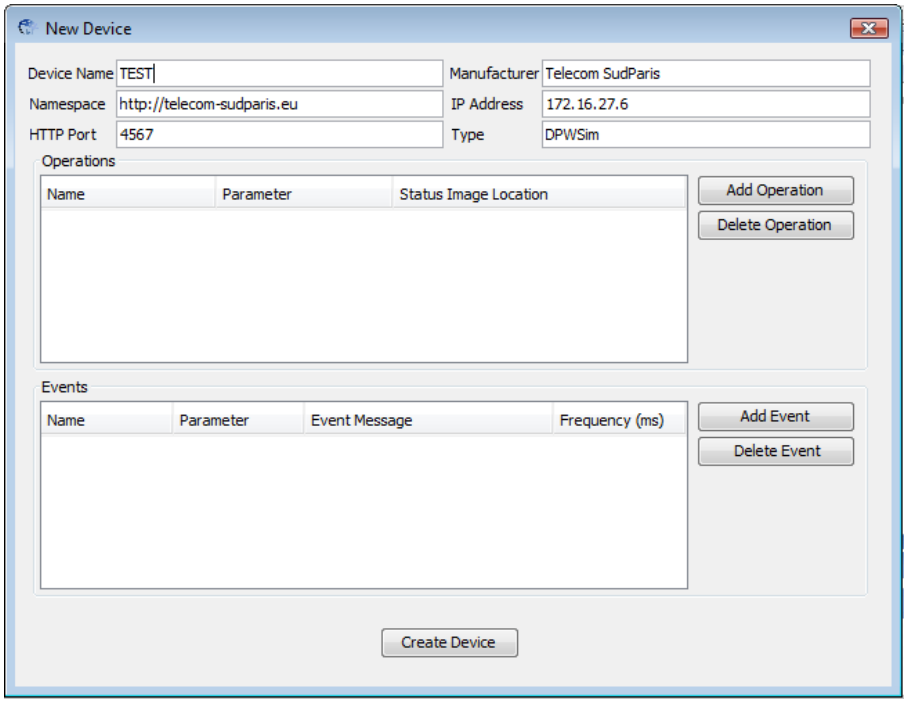
Zadnja verzija: 2.5.5.0 (10.10.2014)

Zrelost programa: začetna

Spletna stran: https://github.com/sonhan/dpwsim

Simulator za prototipiranje, razvoj in testiranje IoT aplikacij z uporabo DPWS.

DPWS je Devices Profile for Web Services. Gre za protokolni sklad na višjih, aplikacijskih nivojih za uporabo SOA arhitekture spletnih servisov na napravah, priključenih na IP omrežje. Včasih ga imenujejo tudi Web Services on Devices (WSD). Vključuje med drugim W3C standarde kot so WSDL in razširjen nabor za naprave, WS-Discovery (odkrivanje naprav v dosegu) in WS-Eventing (prijavo na dogodke naprave, npr. presežena določena mejna vrednost), WS-Security in drugo. Sklad je v osnovi razvil Microsoft (DPWSAPI) in je bil vključen v Windows Vista. S strani različnih razvijalcev je bil dodelan za uporabo na napravah z omejenimi viri. Leta 2009 je bil standardiziran. [77] [78] Pomembnejša iniciativa za uporabo DPWS je evropska WS4D (Web Services for Devices). Za uporabo DPWS so smiselna področja kot so pametni prostori, industrijska avtomatika in avtomobilska tehnologija. [78]



Slika 12: DPWSim

DPWSim simulira poljubno množico virtualnih DPWS naprav v omrežju. Simulator podpira postavitev virtualnih naprav z ločenim naborom operacij in dogodkov. Virtualno DPWS napravo, skreirano z DPWSim, je možno odkriti v omrežju, se naročiti na njene dogodke, uporabljati njene funkcije s strani drugih aplikacij in naprav. S tem je podprt razvoj, prototipiranje in testiranje različnih scenarijev za uporabo arhitekture spletnih servisov na napravah. [79]

Pri uporabi je opaziti kar nekaj pomanjkljivosti, npr. ni podprto kreiranje naprave na podlagi obtoječe, ni možno spreminjati konfiguracije vstavljene naprave in podobno.

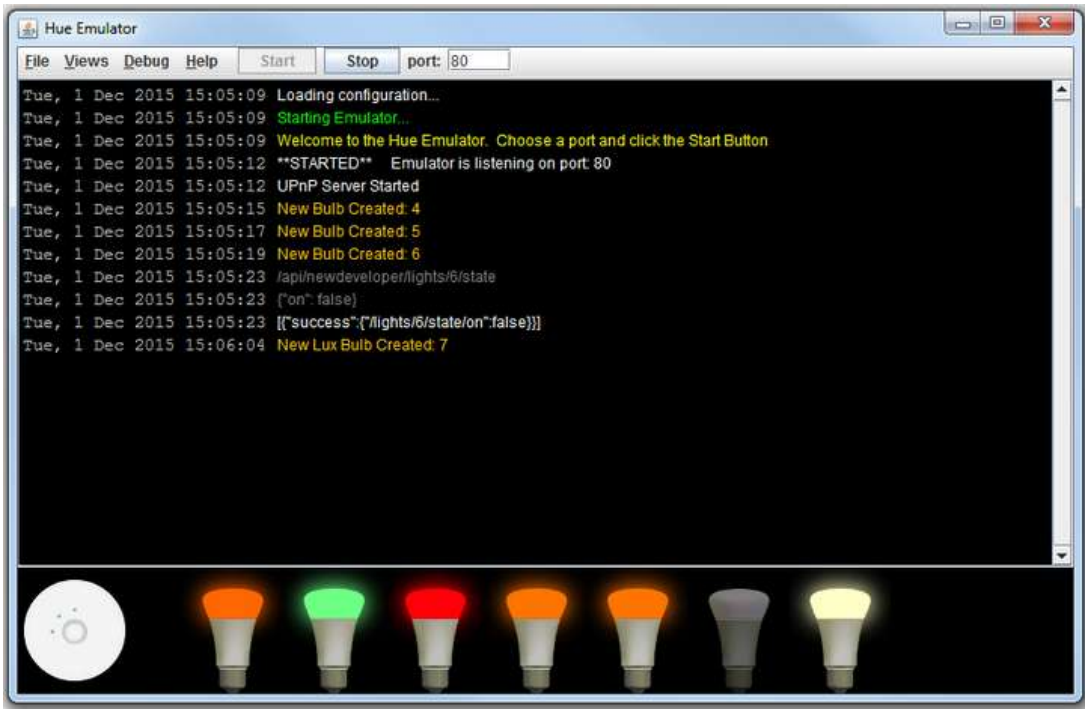
3.1.3.1.1 DPWS Explorer

DPWS Explorer je pomožno orodje za odkrivanje DPWS naprav v omrežju. Omogoča naročanje (subscribe) na storitve naprav, analizo izvedbe, logiranje ipd. http://ws4d.e-technik.uni-rostock.de/dpws-explorer/

3.1.3.2 Hue emulator

Platforma: Java
Tip orodja: emulator, virtualizator
Namen: IoT - Philips pametna sijalka
Tip licence: GNU GPL
Zadnja verzija: 0.7 (december 2015)
Spletna stran: http://steveyo.github.io/Hue-Emulator/
https://github.com/SteveyO/Hue-Emulator

Emulator pametne sijalke Philips. Virtualizira API za potrebe razvoja in eksperimentiranja.



Slika 13: Hue emulator (vir: http://steveyo.github.io/Hue-Emulator/)

3.1.3.3 AllJoyn Device Simulator

Platforma: Windows 10

Tip orodja: testno orodje, simulator, virtualizator

Namen: AllJoyn aplikacije

Tip licence: MIT

Zadnja sprememba: avgust 2016

Spletna stran: https://github.com/dotMorten/AllJoynDeviceSimulator

AllSeen Alliance je povezava proizvajalcev IoT produktov za razvoj standardov za povezovanje in medsebojno komunikacijo raznolikih IoT naprav oz. storitev.

AllJoyn je platforma za testiranje AllJoyn aplikacij. V osnovni verziji je podprta samo generična IoT sijalka, več drugih naprav bo dodanih.

Slika 14: Alljoyn Device Simulator (vir https://github.com/dotMorten/AllJoynDeviceSimulator)

3.1.4 Simulatorji senzorjev

Nekateri simulatorji vsebujejo posebej modele specifičnih tipov senzorjev, pri drugih je potrebno te naprave vzpostaviti z modeli, obstajajo pa tudi ločeni simulatorji senzorjev in senzorskih podatkov.

3.1.4.1 SensorSimulator

Platforma: Java
Tip orodja: simulator senzorjev, virtualizator
Namen: simulira delovanje določenega tipa senzorja (pospešek, temperatura, kompas)
Tip licence: Apache 2.0
Zadnja sprememba: februar 2014
Spletna stran: https://github.com/openintents/sensorsimulator

SensorSimulator je simulator senzorjev kot so senzorji pospeška, svetlobe, magnetnega polja, tlaka, oddaljenosti od predmetov, rotacije, temperature, bar kode, žiroskop, kompas. Senzor je viden v IP omrežju in na voljo za povezovanje v hibridne platforme (SIL), lahko tudi kot knjižnica za izgradnjo ločene programske simulacijske platforme.

3.1.4.2 tsdbwriter

Platforma: .NET
Tip orodja: generator podatkov
Namen. generira analogne senzorske podatke in jih izvozi v TSDB
Zadnja sprememba: april 2015

Spletna stran: https://github.com/hanuk/tsdbwriter
https://dzone.com/articles/c-tool-simulate-iot-sensor

Orodje tsdbwriter je generator analognih senzorskih podatkov (temperatura, pretok, hitrost vetra, pritisk), ki jih zapisuje v časovno podatkovno bazo (Open TSDB). Vrednosti so naključne znotraj vnesenega območja. Konfiguracija je podprta v JSON formatu. [80]

3.1.5 Simulatorji, emulatorji in virtualizatorji vgradnih naprav

Emulatorji in simulatorji mikrokrmilnikov se uporabljajo za razvoj WSN in drugih vgradnih aplikacij brez potrebe po uporabi fizične naprave. Prav tako lahko te simulatorje povezujemo v simulirana heterogena omrežja, v realistične kompozicije za simulacije večjega števila vozlišč.

Simulatorji/emulatorji za specifične modele mikrokrmilnikov in kartic so ponavadi že vsebovani v razvojnem orodju proizvajalca (npr. ARM DS-5 Development Studio, Atmel Studio).

Obstajajo tudi namenski odprtokodni simulatorji/emulatorji za module, ki se uporabljajo v IoT, npr. Arduino. Te vire bomo navedli v naslednjih podpoglavjih.

3.1.5.1 Atmel AVR

Atmel AVR je RISC mikrokrmilnik, ki podpira aplikacije, ki jih je moč prevesti iz programskega jezika C in je zelo razširjen v vgradnih sistemih in WSN omrežjih. Čip je med drugim vgrajen v cenovno dostopne Arduino module, nekatere od njih bomo navedli.

3.1.5.1.1 ATEMU (ATmel EMUlator) [53]

Platforma: Linux
Tip orodja: emulator
Namen: razvoj vgradnih naprav (mikrokrmilnik AVR)
Tip licence: open source
Zadnja sprememba: januar 2004
Spletna stran: http://www.hynet.umd.edu/research/atemu/

ATEMU je emulator za mikrokrmilnik Atmel AVR ter periferno WSN platformo MEMSIC Mica2 (emulacija LED, omrežne enote, senzorjev itd.). Na emuliranem vozlišču je podprta izvedba TinyOS aplikacij. Dodan je tudi model propagacije signala. Debugger je vključen. Dokumentacija ni na voljo.

3.1.5.1.2 SimulAVR [81]

Platforma: Linux, Windows
Tip orodja: simulator

Namen: razvoj vgradnih naprav (mikrokrmilnik AVR)
Tip licence: GNU GPL v2
Zadnja verzija: 1.0.0 (februar 2012) (+novejše beta različice iz leta 2016)
Spletna stran: http://www.nongnu.org/simulavr/
http://savannah.nongnu.org/projects/simulavr

Simulator za družino mikrokrmilnikov Atmel AVR. Napisan je v C++. V osnovi gre za ogrodje, ki je poljubno razširljivo. Posebnega grafičnega vmesnika ni na voljo.
Podprta je skriptna izvedba simulacije na podlagi Python skript.

3.1.5.1.3 SimAVR

Platforma: Linux in OSX
Tip orodja: simulator
Namen: razvoj vgradnih naprav (mikrokrmilnik AVR)
Tip licence: GNU GPL
Zadnja sprememba: april 2016
Spletna stran: https://github.com/buserror/simavr

Simulator za družino mikrokrmilnikov Atmel AVR. Platformo je možno enostavno programsko razširjati.

3.1.5.1.4 Atmel AVR Simulator

Platforma: Motif / Linux
Tip orodja: simulator
Namen: razvoj vgradnih naprav (mikrokrmilnik AVR)
Tip licence: GNU GPL v2
Zadnja verzija: 1.3.0 (februar 2013)
Spletna stran: http://avr.sourceforge.net/

Manjši simulator za Atmel AVR. Vključen je grafični vmesnik. Podprto je spreminjanje hitrosti takta procesorja.

3.1.5.1.5 Emulare

Platforma: Windows
Tip orodja: emulator
Namen: razvoj vgradnih naprav (mikrokrmilnik AVR in Arduino)
Tip licence: GNU GPL
Zadnja verzija: 1.9 (maj 2012)
Spletna stran: http://emulare.sourceforge.net/

Emulare je v osnovi emulator splošne strojne opreme, trenutno je podprt AVR ATMega, na katerega je možno priključiti različne periferne naprave. Podprt je grafični urejevalnik. Podprto je tudi razhroščevanje za Arduino.

3.1.5.1.6 Arduino Debugger/Simulator

Platforma: Windows (Dev-C++)
Razvijalec: Paulware
Tip orodja: simulator
Namen: razvoj vgradnih naprav (Arduino)
Tip licence: open source
Zadnja verzija: 0.09 (junij 2013)
Zrelost programa: eksperimentalna
Spletna stran: https://github.com/Paulware/ArduinoDebugger/
Arduino Debugger je simulator modulov Arduino. Vključen je intuitivni grafični vmesnik. Postopek simulacije ni ločen od programa, ampak se prevede skupaj z njim (potrebno je Dev-C++ okolje).
Pomanjkljivost je dokumentacija.

3.1.5.1.7 Simuino

Platforma: Linux, Web
Tip orodja: simulator
Namen: razvoj vgradnih naprav (Arduino)
Tip licence: GNU GPL v3
Zadnja verzija: 0.2.0 (maj 2014)
Spletna stran: https://code.google.com/archive/p/simuino/
http://web.simuino.com

Simuino je simulator modulov Arduino UNO in MEGA. Izvaja Sketch-programe. Spletna različica (Webuino) trenutno ni na voljo, je pa za dosegljiva različica za Linux.

3.1.5.1.8 UnoArduSim

Platforma: Windows
Tip orodja: simulator, razvojno okolje
Namen: razvoj vgradnih naprav (Arduino)
Tip licence: brezplačen program, koda ni na voljo
Zadnja verzija: 1.6 (junij 2016)
Spletna stran: https://www.sites.google.com/site/unoardusim/

UnoArduSim je novejši simulator za Arduino modul. Podpira izvedbo Sketch-programov. Podprt je razhroščevalnik. Avtor je dodal več I/O naprav, ki jih lahko vključujemo v izvedbo, npr. različni tipi motorčkov, zvočniki, LED diode in druge analogne naprave...
Koda žal ni na voljo, vendar je program brezplačen. Avtor program pogosto nadgrajuje. Dokumentacija je dobro podprta.

3.1.5.2 ARM

3.1.5.2.1 GNU ARM Eclipse

Odprtokodni razvoj na napravah ARM je podprt s paketom razvojnih vtičnikov za ogrodje Eclipse. Paket se redno posodablja. Med drugim so dodani vključki za razhroščevanje v povezavi s simulatorjem QEMU, orodji CodeRed, OpenCD in drugimi. http://gnuarmeclipse.github.io/

3.1.5.2.2 mbed SDK

mbed SDK je razvojna platforma za kontrolnike ARM Cortex in pripadajoče module. https://github.com/mbedmicro/mbed

3.1.5.3 Texas Instruments

3.1.5.3.1 MSPSim

Platforma: Java
Tip orodja: simulator
Namen: razvoj vgradnih naprav (mikrokrmilnik MSP430)
Tip licence: BSD
Zadnja sprememba: april 2009
Spletna stran: https://sourceforge.net/projects/mspsim/
http://tinyos.stanford.edu/tinyos-wiki/index.php/MSPsim

MSPSim je emulator za mikrokrmilnik Texas Instruments MSP430. Emulator je vključen v Cooja simulator. [53] Podpira IHEX in ELF firmware datoteke, vključena so orodja za razhroščevanje.

3.1.5.3.2 WSim

WSim je emulator za MSP430 mikrokrmilnik, ki je vključen v Worldsens Simulator, opisan v razdelku 3.1.1.12.

3.1.5.4 Android in iPhone

Za najbolj razširjeni platformi pametnih telefonov je na voljo obširna ponudba emulatorjev, kar pa presega okvire tega dela, naj omenimo samo RetroArch

3.1.5.5 Drugo

3.1.5.5.1 EMUL8

Platforma: Linux, Mac
Tip orodja: emulator
Namen: razvoj vgradnih naprav (splošno)
Tip licence: MIT
Zadnja verzija: avgust 2016 (dnevne spremembe)
Spletna stran: http://emul8.org
https://github.com/emul8/emul8

EMUL8 je emulator vgradnih naprav na PC. Zgrajen je na modelu ISS (instruction set simulator). Podpira med drugim ARM Cortex-A in Cortex-M. Tip licence je MIT. Možna je integracija tudi npr. s simulacijsko platformo Cooja.

3.1.6 Simulatorji omrežnih algoritmov

Simulatorji v tej kategoriji se uporabljajo za raziskovanje različnih algoritmov distribuiranega omrežja s pomočjo matematičnih metod.

3.1.6.1 Atarraya

Platforma: Java
Tip orodja: simulator
Namen: razvoj algoritmov za kontrolo topologij v WSN
Tip licence: GNU GPL
Zadnja verzija: 1.3 beta (februar 2011)
Spletna stran: http://www.cse.usf.edu/~labrador/Atarraya

Z ustrezno izbiro algoritmov za kontrolo topologij lahko v senzorskih omrežjih pomembno zmanjšamo porabo energije in s tem razpoložljivost sistema ob zadostni meri povezljivosti. [82]

Atarraya je simulator WSN, specializiran za evaluacijo algoritmov za kontrolo topologij. Orodje vključuje model porabe energije, enostaven model mobilnosti in komunikacije ter številne parametre simulacije. Podprta je distribucija vozlišč po ravnini in solidna vizualizacija. Parametrizacija modela je v celoti pokrita z grafičnim vmesnikom, ki je intuitiven za uporabo. [53] Komunikacija med vozlišči poteka na osnovi sporočil (messaging). Navodila so dobro podprta.

3.1.6.2 AlgoSenSim

Platforma: Java
Tip orodja: simulator, simulacijsko ogrodje

Namen: porazdeljeni algoritmi (WSN)
Tip licence: GNU GPL
Zrelost programa: eksperimentalna
Zadnja verzija: 0.9.2.2 (september 2006)
Spletna stran: http://tcs.unige.ch/doku.php/code/algosensim/overview
Dokumentacija: http://tcs.unige.ch/javadoc/algosensim/

AlgoSenSim je simulacijsko ogrodje / simulator za algoritme, ki se uporabljajo v splošnih distribuiranih omrežjih, npr. za lokacijo vozlišča, usmerjevalne algoritme, poplavljanje s sporočili. V instalacijskem paketu je dodatnih več konkretnih primerov takšnih algoritmov. Komunikacija med vozlišči poteka na osnovi sporočil.

Model vzpostavimo s pomočjo XML konfiguracijskih datotek. Simulator je zgrajen modularno, ločena je izvedba simulacije od reprezentacije. Vizualizacija izvedbe je omogočena. [53] Dokumentacija je dobro podprta.

Orodje je uporabno za simulacijo algoritmov v topologijah WSN, ad hoc topologijah ipd.

3.1.6.3 Sinalgo

Platforma: Java
Ustanova: Swiss Federal Institute of Technology Zurich
Tip orodja: simulator
Namen: omrežni algoritmi
Tip licence: BSD
Zadnja verzija: 0.75.3 (april 2008)
Spletna stran: http://disco.ethz.ch/projects/sinalgo/

Sinalgo je simulator omrežnih algoritmov. Omrežna simulacija je rešena aproksimativno (podobna strategija kot npr. Shawn), s tem so avtorji dosegli skalabilnost vozlišč v rangu 100.000. Komunikacija med vozlišči je vzpostavljena na osnovi sporočil. Vključeni so različni modeli za mobilnost, komunikacijo, prenos, distribucijo (postavitve vozlišč), interferenco ter izgubni prenosni kanal. Konfiguriranje je podprto z grafičnim vmesnikom in XML konfiguracijskimi skriptami, logika na vozliščih pa v jeziku Java. Omogočena je grafična vizualizacija izvedbe v živo.

3.1.7 Operacijski sistemi

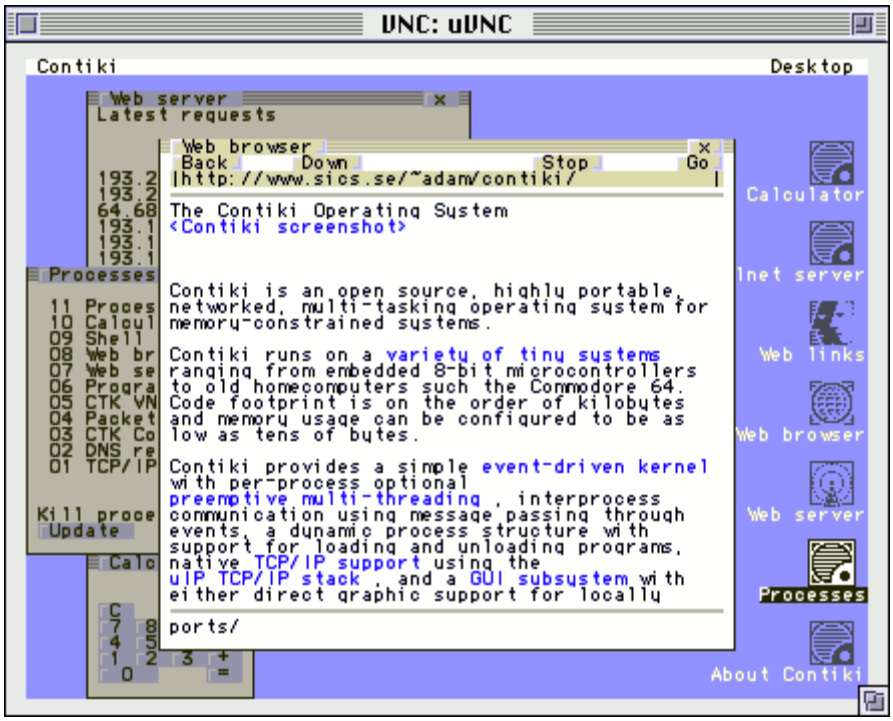
Navedli bomo simulacijske platforme, neposredno vezane na določene operacijske sisteme za vgradne naprave oz. operacijske sisteme realnega časa.

3.1.7.1 Contiki / Cooja

Platforma: C, Linux
Razvijalec: Thingsquare (Švedska), prej SICS

Tip orodja: Contiki - operacijski sistem, Cooja - simulator
Namen: naprave z omejenimi viri (IoT / WSN)
Tip licence: BSD open source
Zadnja verzija: 3.0 (25.8.2015)
Spletna stran: http://www.contiki-os.org
Razširitve osnovnega sistema: https://github.com/contiki-os/contiki/wiki

Contiki in Cooja sta med prvimi zadetki, ki jih najbolj znani iskalnik vrne pod nizom »IoT simulator«. Contiki je odprtokodni operacijski sistem za naprave z omejenimi viri (IoT naprave / WSN). Nameščen na končno napravo zaseda manj kot 2kB pomnilnika in podpira multi tasking (event-based). [83] Contiki je podprt na množici naprav kot so družine naprav ARM, AVR, TI SensorTag. Pri implementaciji sistema so sodelovala podjetja kot Atmel, Cisco, ETH, Redwire LLC, SAP in Thingsquare.

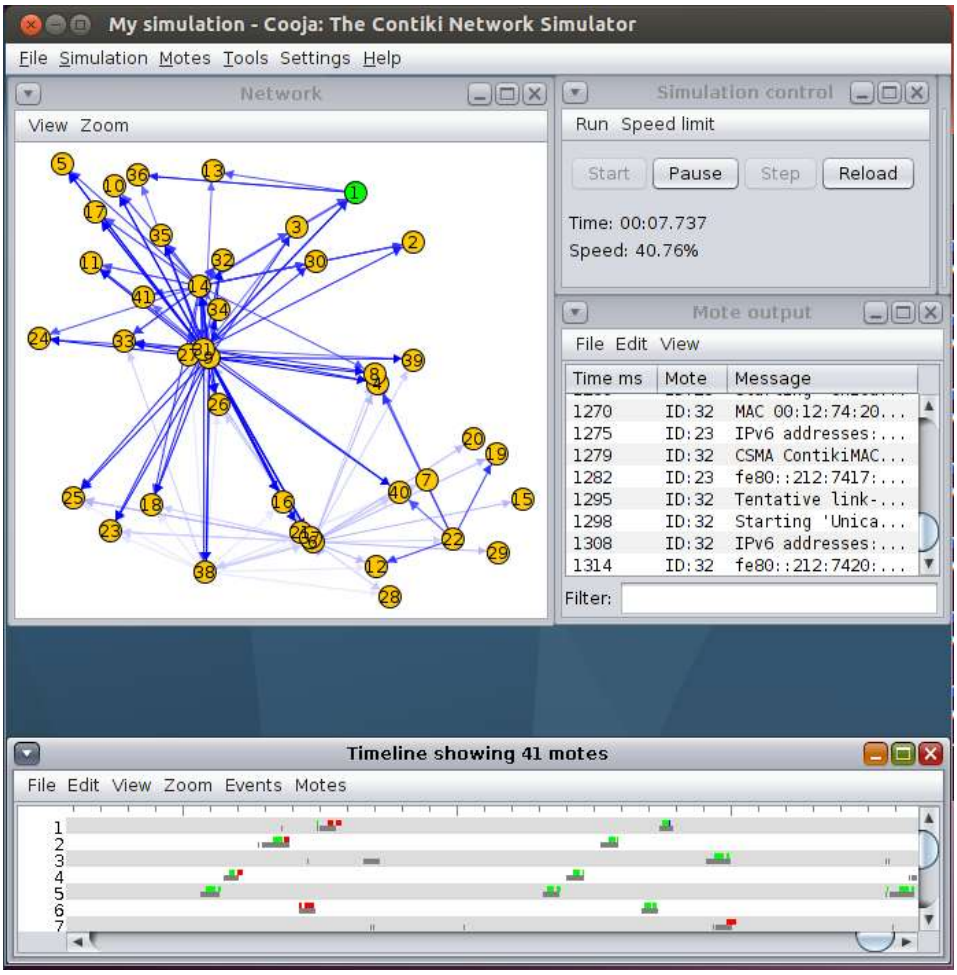


Slika 15: Contiki (vir Wikipedia)

Na podlagi Contiki je bila zgrajena simulacijska platforma / grafično razvojno oz. orodje Cooja. Orodje je v pomoč pri prototipiranju, razvoju in testiranju rešitve pred namestitvijo na končne naprave.

Programska koda, ki teče v simulaciji, je enaka tisti na napravah - ni potrebno npr. prevajanje iz skriptnega jezika v končno programsko rešitev. To je prednost pred domenskimi simulatorji, ki jo izkoriščajo razvojna orodja, vezana na operacijski sistem. Cooja podpira manjši delež naprav, ki jih sicer podpira Contiki. Torej v teh primerih je emulacija izbrane naprave možna. Seznam podprtih naprav je osvežen na spletni strani proizvajalca.

Nivo višje je implementiran tudi poseben skriptni jezik za avtomatsko izvedbo simulacijskih postopkov.



Slika 16: Cooja (vir: Wikipedia)

Za simulacijo radijskih povezav je v celoti podprt IPv4 in IPv6, torej UDP, TCP in HTTP. IPv6 sklad certificiran, dobavljen s strani Cisco Systems. Za naprave z omejenimi viri so podprti še standardi 6LowPAN, RPL, CoAP. Z verzijo Contiki 3.0 je podprt tudi standard MQTT. Za enostavnejše postavitve je sicer možno uporabiti tudi posebni priloženi Contiki-jev protokolni sklad Rime.

Simulacija porabe energije je podprta z energetskim modelom Energest. Analiza tega nivoja je podprta z orodjem Powertrace ter vtičnikom PowerTracker. [84]

Orodje Cooja je možno povezati na NodeRED. [83]

V osnovi sistem ne podpira določenih funkcionalnosti, ki jih najdemo pri drugih platformah, npr. model za mobilnost. Arhitektura vtičnikov pa omogoča dodajanje poljubnih funkcionalnosti, ki jih uporabniki objavljajo na github: strojna podpora za Arduino, Senslab, BatMote ipd., vtičniki za povezavo z SQLite bazo, trace orodje, podporo za mobilnost,

GISOO (povezava Cooja in Simulink), vtičnik za pomoč pri namestitvah programske kode na končnih napravah (Cooja TWIST).

Simulator je močan v smislu emulacijske podpore točno določenim napravam za rang IoT. Sistem se pogosto uporablja v študijskih programih, relativno veliko je prezentacij na youtube.

Slabost je skromna dokumentacija. V praksi tudi ni najbolj produktiven. [83] Zraven tega bi še dodali, da v vizualizaciji ni možno prikazati konteksta kot je mesto ali prostor, simulacija mobilnosti je osnovna po vnaprej začrtani poti, ni podprta npr. z GPS podatki, ki bi jih lahko uvozili ali posredovali iz zunanjega simulatorja ali priključene naprave. Simulator je namenjen za programsko testiranje za splošni kontekst. Ne podpira simulacije večjega števila vozlišč.

3.1.7.2 RIOT

Platforma: C, C++, podporna orodja Linux
Podporne ustanove: Freie Universität Berlin, INRIA, Hamburg University of Applied Sciences
Tip orodja: operacijski sistem
Namen: naprave z omejenimi viri (IoT)
Tip licence: LGPL
Zadnja sprememba: avgust 2016 (dnevne spremembe)
Spletna stran: http://www.riot-os.org/
https://github.com/RIOT-OS/

Podobno kot pri Contiki, gre za odprtokodni operacijski sistem za naprave z omejenimi viri, omogočena je večopravilnost. Podpira različne strojne arhitekture ARM7, MSP430, ARM Cortex (M0, M3 in M4), kot tudi različne mikrokrmilnike oz. module (Arduino, Atmel, mbed, Nordic, OpenMote, Airfy, Telos, Texas Instruments) - točen seznam podprtih modelov kot tudi podprtih gonilnikov za različne tipe senzorjev je objavljen na spletni strani. Podprti so protokoli 802.15.4, 802.15.4e, IPv6, 6LoWPAN, RPL, NHDP, AODVv2, CCN-lite, TCP, UDP, CoAP, CBPR.

Od drugih operacijskih sistemov se razlikuje v tem, da podpira programsko kodo direktno v jeziku C++. Razvoj lahko poteka na PC brez prisotne fizične končne naprave, podprta sta Linux in BSD/Mac OS. Omogočena je tudi virtualizacija in povezovanje več RIOT instanc v testno okolje. Ker gre za operacijski sistem, je možna izvedba enake kode na različnih podprtih napravah. Podprta pa je nenazadnje tudi namestitev kode na končno napravo.

Obstajajo metode, kako RIOT povezati s simulatorjem Cooja. [85]

Za bodoče napovedujejo podporo za protokol BLE. [86]

Sistem je podprt z dokumentacijo, skupnost je aktivna.

3.1.7.3 TOSSIM

Platforma: C++, Python
Ustanova: Stanford University
Tip orodja: simulator
Namen: razvoj vgradnih naprav za TinyOS
Tip licence: BSD
Spletna stran: http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM
https://sensorweb.cs.gsu.edu/wiki/index.php/TOSSIM

TOSSIM je simulator za TinyOS aplikacije. Podpira WSN platformo MicaZ. [53] Programska koda je napisana v jeziku C. Za simulacijsko skripto se uporablja C++ in Python. [87] Vizualizacija je podprta z orodjem TinyViz, ki med drugim omogoča izvajanje več hkratnih simulacij, razhroščevanje, zbiranje statistike ipd.

Težko je detektirati napake pri poteku simulacije. Ni vključen model okolja in porabe energije, vključuje samo verjetnostne približke napak pri prenosu. [88] [53] Zadnja verzija programa je iz leta 2008. Novejše platforme in naprave so slabo podprte, na voljo pa so dodatki za simulator, npr. SimX ali Tython [53] ter razširitve, npr. PowerTOSSIM (za realistično evaluacijo energetskih modelov) [89] in pa SUNSHINE (integracija TOSSIM, SimulAVR in GEZEL). [53] Kljub temu je to razširjen in uporabljan simulator, ravno zaradi navezave na TinyOS.

Alternativa za simulacijo TinyOS aplikacij je sicer še NesCT, ki je razširitev OMNeT++.

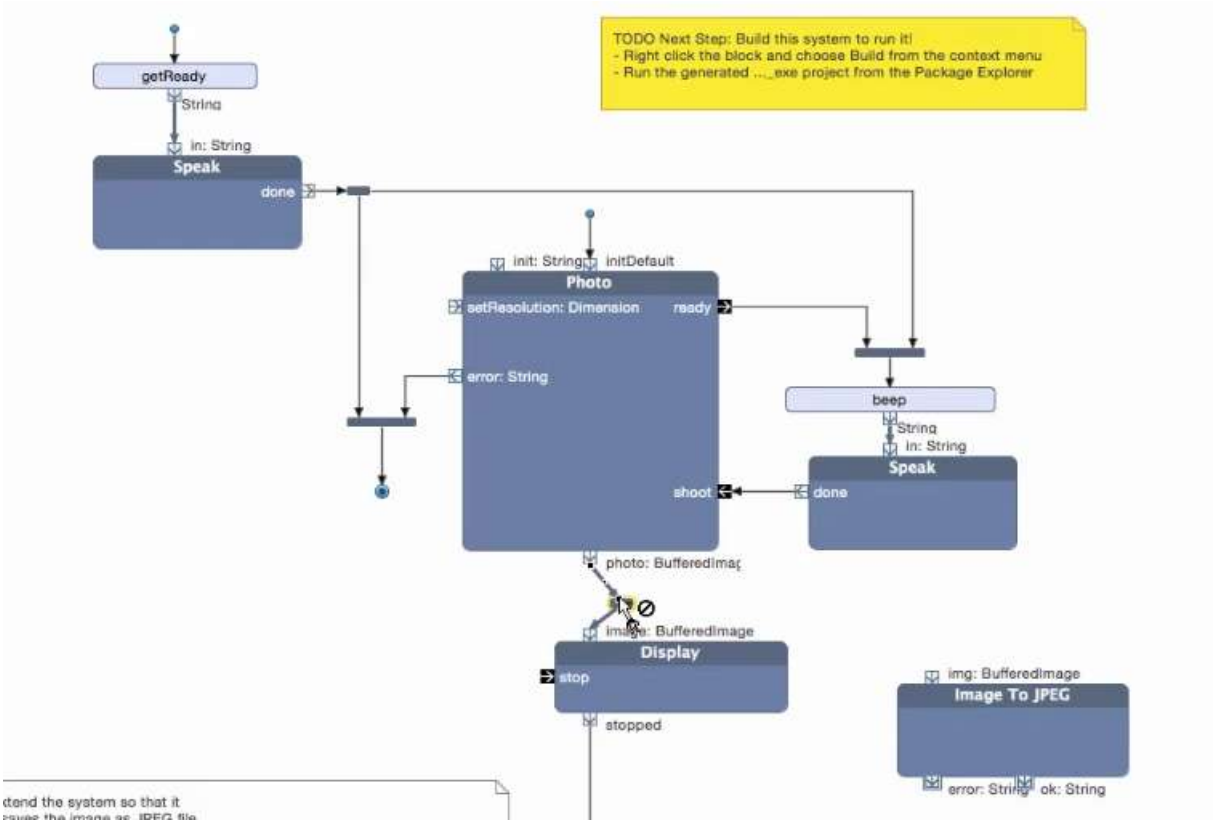
3.1.8 Razvojna in prototipna orodja

3.1.8.1 Reactive Blocks [90]

Platforma: Java (Eclipse)
Ustanove: bitreactive (pred tem Department of Telematics / Norwegian University o Science and Technology)
Tip orodja: orodje za prototipiranje, razvojno okolje
Namen: razvoj aplikacijskih rešitev za medmrežne vmesnike (IoT gateways)
Tip licence: EPL
Zadnja verzija: 2.6.0 (nov 2015)
Spletna stran: http://www.bitreactive.com

Reactive Blocks je orodje za razvoj IoT s pomočjo gradnih elementov (building blocks). Navajamo ga zato, ker omogoča modeliranje / prototipiranje sistema.

Programsko rešitev implementiramo s povezovanjem vnaprej pripravljenih kock - elementov. Vsak element ima definiran specifični vmesnik, preko katerega ga povezujemo z drugimi elementi (različica API). Posamezni elementi vsebujejo kose kode, ki se izvajajo.



Slika 17: BitReactive (vir: http://www.bitreactive.com)

Sintaksa modela je v osnovi UML. S tem je omogočena večnivojska objektna zasnova ter verifikacija modela (model check), kar je npr. uporabno za detekcijo mrtvih zank ipd.

Če je koda konsistentna, se generira OSGi paket in ga namesti na končno napravo, lahko kot samostojna Java aplikacija.

Na voljo so naslednji elementi:

- serializacija: GSON, DOM4J,
- komunikacija: MQTT, HTTP , HTTPS, Email, Xively, XMPP, Twilio SMS, Keyteq SMS, AMQP, OPC-UA, LoRa,
- fizična komunikacija: Modbus, Serijski vhod / izhod, RPi GPIO Digital,
- ravnanje z datotekami,
- mobilnost - geofence, KML,
- prototipiranje - Java Swing, text-to-speech in
- drugo: USB kamera (detekcija gibanja), procesiranje video signala.

Razvijalec lahko uporabi osnovne gradnike, lahko pa uporabi že zgrajene vzorce kode, ki so objavljeni online na strani proizvajalca.

3.1.8.2 CodeBlocks Arduino IDE

Platforma: Windows, Linux; C++
Tip orodja: integrirano razvojno okolje (IDE)
Namen: razvoj vgradnih naprav (za modul Arduino)
Tip licence: GNU GPL
Zadnja verzija: 1.0b2 (junij 2014)
Spletna stran: http://arduinodev.com/codeblocks/
https://github.com/provideyourown/CodeBlocks-Arduino

CodeBlocks Arduino IDE je produktivno razvojno okolje za aplikacije za modul Arduino. Vključuje urejevalnik kode (sketch-programov), orodja za namestitev na Arduino napravo in druga pomožna orodja. Podprt je tudi simulator, ki je še v začetni razvojni fazi.

3.1.8.3 EvoThings Studio

Razvijalec: Evothings
Tip orodja: razvojno in prototipno orodje
Namen: za mobilne aplikacije v povezavi z IoT
Tip licence: Apache
Zadnja verzija: 2.1.0 (13.6.2016)
Spletna stran: http://evothings.com/
Izvorna koda: https://gitter.im/evothings/evothings

Slabost razvoja mobilnih aplikacij je zahtevno in zamudno nameščanje programske opreme in popravkov na naprave. Evothings Studio je orodje za hiter razvoj mobilnih aplikacij za IoT. Gre za neke vrste virtualno okolje, ločeno od operacijskega sistema mobilne naprave. Na podprtih napravah namestimo odjemalec Evothings Viewer, ki izvaja aplikacije. Aplikacije so v osnovi spletne (uporablja se HTML, Javascript in CSS), vendar Evothings Viewer ni spletni brskalnik. Omogoča večjo kontrolo naprave na daljavo - instantno distribucijo popravkov. [91] Za ta namen se uporablja orodje Evothings Workbench, ki vključuje možnost preproste distribucije popravkov na končne mobilne kliente (push). S tem je omogočen razvoj in prototipiranje kar neposredno na napravah.

Napisane aplikacije je s pomočjo Apache Cordova možno tudi prevesti v kodo ciljnega operacijskega sistema (Android, iOS). Vhodna koda pa je enaka kot pri uporabi vmesnika Evothings Viewer. Preko Cordova so podprti standardi BLE (Bluetooth Low Energy), HTTP, TCP, UDP, iBeacon in drugo.

Podprta je komunikacija z IoT mikrokrmilnikih kot so so Arduino, ARM mbed, RedBearLab, LinkIt One, ESP8266, TI SensorTag. Podprt je med drugim tudi Raspberry Pi. Potrebno je poudariti, da EvoThings ni middleware niti razvojno orodje, ki bi olajšalo razvoj programske opreme za same mikrokrmilnike. Omogoča hitro izdelavo mobilnih prototipov, ki z IoT

neposredno komunicirajo. Programsko kodo na IoT mikrokrmilnikih je potrebno napisati v specifičnih razvojnih okoljih (mbed npr.) ali orodjih v sklopu operacijskih sistemov Riot in Contiki, TinyOS ipd.

Dokumentacija vsebuje obširne opise za povezovanje podprtih naprav v aplikativno rešitev, skupaj s primeri.

3.1.8.4 WComp

Platforma: .NET, C#
Ustanova: Universite Nice Sophia Antipolis
Tip orodja: orodje za prototipiranje, middleware
Namen: vseprisotno računalništvo, IoT
Zadnja verzija: 3.2.1 (november 2014)
Zrelost programa: prototip
Spletna stran: http://www.wcomp.fr/

Orodje za prototipiranje in izvedbo (middleware) mobilnih distribuiranih aplikacij (aplikativni nivo) s ciljem skrajšanja razvoja. Komunikacija med napravami poteka preko spletnih servisov, konkretno DPWS in UPnP. Načrtovanje aplikacije je podprto z grafičnim vmesnikom (designers). Orodje je možno uporabljati tudi na napravah kot je Raspberry Pi, Android naprave itd. Zgrajen je na osnovi ogrodja/modela SLCA.

Žal od leta 2014 ni posodobitev, čeprav so bile napovedane. Dokumentacija je osnovna in v francoskem jeziku.

3.1.9 Druga orodja za podporo simulacijam

3.1.9.1 wotio/ripple

Orodje wotio/ripple je generator MQ sporočil. Možno je nastaviti frekvenco oz. bolj podrobne vzorce porajanja sporočil v času. Podprto je nastavljanje topologije. Vir [92] poveže simulator na Rabbit MQ in prikazuje vizualizacijo in analizo.

Orodje je uporabno za evaluacijo messaging funkcionalnosti v M2M aplikacijah, posebej npr. za testiranje obremenitvenih zmogljivosti sistema. Orodje bi bilo smiselno dodelati za generiranje MQTT.

https://github.com/wotio/ripple

3.1.9.2 .NET Gadgeteer

Platforma: .NET
Razvijalec: Microsoft
Tip orodja: platforma za hitro prototipiranje, standard za vgradne naprave
Namen: vgradne naprave

Tip licence: ASL, CC
Zadnja verzija: 2.44.11 (november 2015)
Spletna stran: http://www.netmf.com/gadgeteer/
https://gadgeteer.codeplex.com/

Platforma (razvojni paket) za hitro prototipiranje vgradnih naprav. Vključuje naprave posebne vgradne module, standardizirane za middleware platformo .NET Micro Framework (NETMF). Na voljo so različne naprave (senzorji, kamere, I/O naprave ipd.), ki jih povezujemo na mikrokrmilnik, na katerem teče NETMF. Naprave se priključujejo s plug&play. Programiranje modulov je podprto neposredno s C# .NET v orodju Visual Studio.

Pri Microsoftu so platformo razvili za potrebe lastnega raziskovanja in razvoja. Zaradi zanimanja v skupnosti pa so platformo ponudili javnosti kot open source. [93] Platforma omogoča hiter razvoj vgradnih prototipov v RAD okolju na ogrodju .NET, po produktivnosti se zaradi tega bolj približa razvoju programske opreme za PC.

3.1.9.3 Automate

Platforma: Android
Tip orodja: izvajalec opravil na osnovi android dogodkov
Namen orodja: RAD, prototipiranje, M2M, mobilne aplikacije
Tip licence: zasebna, prosti dostop
Zadnja verzija: 1.3.0 (avgust 2016)
Spletna stran: http://llamalab.com/automate/

Automate je aplikacija za avtomatizacijo opravil v sistemu Android. Na osnovi določenih pogojev (senzorji in dogodki) prožimo poljubne akcije. Na voljo je knjižnica z elementi (pogoji in akcijami), ki jih povezujemo med seboj. Urejanje postopka je podobno risanju končnega avtomata, deloma tudi Petrijevih mrež (npr. izvedba pogoja AND z dvema ločenima elementoma, pri čemer morata biti oba izpolnjena - princip »žetonov«).

Pogoji so npr. vezani na senzorje (žiroskop, GPS, termometer, senzor svetlobe ipd.), dogodke (čas, ponovni zagon, prejeta e-pošta, zaznava NFC). Akcije so po drugi strani npr. izvedbe klasičnih aktuatorjev (proženje zvoka, aktivacija ekrana) kot tudi različne operacije (HTTP ali REST klic, FTP, pošiljanje SMS, e-pošte, aktivacija aplikacije, snemanje zvoka itd.), ki jih Android omogoča.

Ne glede na to je s tem orodjem možno podpreti najrazličnejše praktične scenarije, za katere bi s pomočjo razvojnega orodja potrebovali več časa, prevajanja itd., nenazadnje pa je zaradi vizuelnega orodja bližje širši množici uporabnikov, ki lahko brez potrebe po znanju programiranja sami izdelajo enostavne operacije. Gre torej za alternativo razvojnemu orodju, kar je zanimivo za razvoj / prototipiranje enostavnih IoT aplikacij (WSN in M2M).

Primeri uporabe: pošlji SMS, ko se telefon poveže v točno določeno GSM celico; pošlji e-pošto, ko se bluetooth poveže na točno določeno napravo, kliči funkcijo določene druge naprave, če se telefon obrne na glavo; izklopi zvok, če zazna določeno NFC napravo. Omogočeno je upravljanje tudi z drugimi aplikacijami preko vključkov, kompatibilnih s Tasker ali Locale, npr. povezava z vremenskimi dogodki. Preko API-jev je možen npr. scenarij prižgi HI-FI, ko se telefon pojavi v domačen WI-FI omrežju, prižgi snemalnik zvoka na določeni GPS lokaciji in podobno. Vse kompatibilne aplikacije, ki jih je možno upravljati, so zbrane na tem naslovu: http://tasker.wikidot.com/plug-ins-and-3rd-party.

Za določene izvedbe je realizacija težja (npr. enostavni logični predikati kot sta operaciji OR ali XOR) je potrebnih več izključujočih gradnikov in izvedbo zakomplicira. Tudi ponovna uporaba algoritmov ni najbolje podprta (manjka objektnost).

Vzpostavljena je sicer tudi knjižnica, ki jo vzdržuje skupnost, ki objavlja praktične rešene postopke. Algoritem iz knjižnice je na enostaven način možno poiskati, prevzeti in namestiti na napravo.

Program je sicer prostodostopen z določenimi omejitvami, vendar zadošča za osnovne postopke za podporo enostavnih primerov uporabe, polna različica pa je na voljo za simbolično ceno.

3.1.9.4 DomoNet

Platforma: Java
Tip orodja: middleware, ontologija
Namen:vseprisotno računalništvo - pametni prostor
Tip licence: GNU GPL 2
Zadnja sprememba: november 2009
Spletna stran: http://domonet.sourceforge.net/
https://sourceforge.net/projects/domonet/

Odprtokodno modularno programsko ogrodje / vmesno programje za področje pametnih prostorov. Vmesno programje (middleware) poenoti način komunikacije med napravami različnih proizvajalcev in standardov. Uporablja skupni označevalni jezik za pametne prostore - DomoML. [94]

3.1.9.4.1 DomoML

DomoML je jezik za DomoNet. V osnovi je to ontologija s področja vseprisotnega računalništva, konkretno za pametne domove. Ontologije je smiselno upoštevati pri izgradnji simulatorja za določeno domeno, prevajalnikov v končno kodo naprav ipd. Podobna ontologija za pametni dom je tudi DogOnt. Primer ogrodje za razvoj sistemov pametnih prostorov je AMiGO. Druge ontologije, primerne za dano področje, so še SOUPA, Cobra-ONT, GAIA, Gas, Conon [95].

3.1.9.5 Eclipse IoT

Eclipse ponuja IoT ogrodje, pod katerim združuje več odprtokodnih rešitev za razvoj IoT. Ti produkti so odprtokodni in bi jih bilo smiselno uporabiti za SIL.
kura - open source OSGi ogrodje za IoT medmrežni vmesnik (npr. Raspberry Pi)
paho - MQTT klient
Mosquitto - MQTT server
eclipse smarthome - ogrodje za pametne domove, ki se razširja v končne produkte kot je npr. middleware openHAB.
Leshan - OMA LWM2M server
Californium - ogrodje za podporo CoAP
Celoten spisek projektov Eclipse IoT: http://iot.eclipse.org/projects

3.1.9.6 Ogrodje za testiranje WoT

Platforma: C#
Tip orodja: testno okolje
Namen: Web of Things
Zadnja sprememba: april 2016
Izvorna koda: https://github.com/MinaYounan-CS/WoT-Testbed-Environment

Web of things (WoT) se osredotoča na aplikacijski nivo IoT. Younan, Khattab in Bahgat [96] predlagajo testno okolje za WoT, ki bi poenotilo različne spletne formate in omrežno infrastrukturo ter omogočilo indeksiranje spletnih točk na spletnih iskalnikih. Vozlišča komunicirajo preko RESTful spletnih servisov. Za osveževanje statusov senzorjev je uporabljen AJAX.

3.1.9.7 EXEHDA-AD

Middleware za podporo dinamičnim adaptivnim sistemom na področju vseprisotnega računalništva. [97]
Izvorna koda: https://sourceforge.net/projects/exehdaad/

3.1.9.8 Node-RED

Vizualno orodje za povezovanje API-jev in drugih storitev v aplikacijo.
http://nodered.org/

3.1.10 Raziskovalni projekti in metodologije

Na področju simulacij za IoT smo naleteli na nekatere raziskovalne projekte brez konkretne razvite programske opreme, vendar izpostavljajo teoretične arhitekturne zasnove, ontologije in druge koncepte na področju simulacij, ki bi bile potencialno uporabne za nadaljnji razvoj simulatorjev.

3.1.10.1 Dinamično vizualno simulacijsko orodje za IoT [12]

Platforma: C# .NET
Ustanova: Sparks - I3S - Sophia Antipolis
Tip: predlog ogrodja
Namen: pametni prostori, IoT
Datum objave: avg 2015
Spletna stran: http://sparks.i3s.unice.fr/
Status: v razvoju

Razvojna ekipa Sparks predlaga [12] vzpostavitev novega simulacijskega ogrodja za učinkovito simulacijo pametnih prostorov. Programska oprema bi vključevala naslednje kriterije: a) 3D grafično orodje za vizualizacijo in modeliranje scenarijev, b) enostavno dodajanje novih naprav brez potrebe po modifikaciji 3D grafičnih rutin, c) šibka sklopljenost med grafičnimi rutinami in simuliranimi storitvami d) upravljanje s senzorji in aktuatorji ter podpora namestitvi v končno okolje, e) podpora za avtomatsko odkrivanje (discovery) naprav in storitev glede na kontekst (npr. z vpeljavo spletnih storitev za naprave in UPnP protokola), f) spremembe na modelu (konfiguraciji naprav in storitev) med izvajanjem simulacije.

Omogočili bi tudi hibridne simulacije (uporaba simuliranih in fizičnih naprav v modelu). V članku so predstavljene primerjave tudi z drugimi simulatorji, ti simulatorji pa žal niso dostopni, da bi jih preverili.

Omejitev testnega sistema iz članka je 250 naprav. Avtorji navajajo, da bosta program in koda prosto dostopna. Platforma je C# .NET (mono), knjižnica Intel UPnP, za 3D vizualizacijo pa se uporablja ogrodje Unity 4.5.5..

3.1.10.2 Metodologija za skalabilne simulacije [29]

Platforma: Java
Ustanova: Department of Information Engineering, University of Parma, Italija
Tip: metodologija
Namen: pametna mesta, veliko število vozlišč (rang 100.000 vozlišč, nekaj 100 milijonov dogodkov)
Datum objave: oktober 2014

V članku [29] je predlagana metodologija in simulacijska platforma za obsežne IoT scenarije (večje število vozlišč), predvsem za pametna mesta. Izpostavljene so določene slabosti simulacijskih okolij, ki natančno emulirajo delovanje vozlišč in protokolnega sklada (npr. Cooja ali TOSSIM). Zaradi natančnosti izvedbe ta orodja niso optimizirana za izvedbo skalabilnih modelov. Avtorji predlagajo vpeljavo splošnega simulatorja in uporabo enostavnejših modelov, s čemer se sprostijo resursi in omogoči skalabilnost.

Ključni kriteriji uspeha, ki jih navajajo avtorji, so ponovna uporaba kode, intuitivna uporaba orodja, modularna zasnova, enostavna namestitev.

Za simuliranje obsežnih IoT scenarijev predlagajo naslednje korake. 1. Identificiraj glavne podsisteme glede na posebne komunikacijske značilnosti. 2. Implementiraj posamezne podsisteme v natančnih simulatorjih kot so npr. Cooja ali ns-3 ter izvedi evaluacijo omrežne komunikacije in porabe energije. 3. Prevedi dobljene ugotovljene rezultate posameznih podsistemov v model splošnega simulatorja DEUS in izvedi simulacijo celotnega distribuiranega okolja.

Splošno ogrodje DEUS smo sicer predstavili v poglavju 3.1.1.16.

3.1.10.3 DiaSuite

Platforma: Java
Ustanova: Phoenix, Inria Bordeaux, Francija
Tip orodja: razvojno in simulacijsko orodje
Namen: pametni prostori, prodorno računalništvo
Datum objave: november 2015
Status projekta: raziskovalni projekt, trenutno ocenjujejo možnosti za uporabo v industriji [33]

Orodje za zasnovo, implementacijo, testiranje (simulacijo), namestitev in izvedbo IoT aplikacij. Podpira celotni razvojni cikel, torej gre za celostno razvojno orodje za IoT. V model je možno vključiti senzorje (svetlobni senzor, pozicija, detekcija gibanja...), aktuatorje (luč, ključavnica, zvonec), agente (premikanje ljudi po prostoru), kontekst (vizuelna upodobitev scenarija - stavba, mesto) ter fizične dejavnike (temperatura, vlaga). [98] Definirati pa je možno tudi poljubne nove naprave.

Za modeliranje je podprt grafični vmesnik, na voljo pa je tudi visokonivojski skriptni jezik DiaSpec. Dodan je repozitorij končnih naprav in senzorjev, kar poenostavi uporabo. [99]. Vključene so rutine za pretvorbo skriptne kode v končni jezik naprav. S tem je podprta namestitev na fizičnih napravah.[100] Vizualizacija izvedbe je podprta z orodjem Siafu, ki smo ga opisali v poglavju 3.1.2.2.

Pokriva višji nivo aplikacijske uporabe, zato ne pokriva simulacij omrežnega prometa. Ni podprto parametriranje vedenja ljudi v modelu. [28] Ni namenjen za simuliranje velike množice naprav in senzorjev, za ta namen Phoenix razvija produkt DiaSwarm [101]. Avtorji so sicer projekt DiaSuite zaprli za javnost in iščejo možnosti za komercialno uporabo. [33]

3.2 Primerjalna tabela med zbranimi orodji

Na osnovi opisnih in drugih primerljivih parametrov bomo izvedli primerjavo med orodji, ki smo jih zbrali v prejšnjem poglavju. Primerjavo smo oblikovali sistematično in pregledno, da

bi strokovnjakom iz različnih panog olajšali preiskovanje po simulacijskih orodjih za točno določeno fazo projekta IoT. V primerjavo smo vključili le tista orodja, o katerih je bilo možno pridobiti zadosti informacij. Tabela služi za začetno iskanje, zahtevnejši bralec pa bo več informacij našel v prejšnjem poglavju.

3.2.1 Tabela I - Osnovna primerjava

Naredili bomo osnovno primerjavo med orodji po osnovnih opisnih kriterijih. Stolpci so naslednji (zaloga vrednosti je navedena v *kurzivi*):

- Naziv orodja;
- Domena IoT - aplikativno področje IoT, ki ga orodje pokriva (*pametno mesto, povezani promet...*). Vrednost *[ni omejitev]* pomeni, da se lahko orodje uporablja za zelo različne domene;
- Funkcionalni opis - opis osnovnih funkcionalnosti in značilnosti orodja. Podrobnejši opis orodja je vsebovan v prejšnjih poglavjih;
- Namen uporabe v IoT - kako in v katerih fazah razvoja IoT si lahko pomagamo z orodjem;
- Nivo sistema - navedemo področja sistema, na katere se simulacijska platforma osredotoča:
    - *aplikativni* - poslovna pravila in programska oprema distribuirana po vgradnih napravah, medmrežnih vmesnikih, oblaku,
    - *arhitekturni* - strojne in programske komponente sistema in hierarhija med njimi,
    - *komunikacijski* - komunikacijsko omrežje, topologija sistema,
    - *varnostni* - varnost sistema,
    - *kontekst* - modeli dinamike stimulusov, reprezentacija izvedbe na grafičnem ozadju;
- Postopki modeliranja in izvedbe - na kratko opišemo osnovno uporabo oziroma usmerimo uporabnika, kje je najbolje začeti.

Tabela I - osnovna primerjava

Naziv orodja	Domena IoT	Funkcionalni opis	Namen uporabe v IoT	Nivo sistema	Postopki modeliranja in izvedbe
Simulatorji					
UbiREAL	pametni prostori, vseprisotno računalništvo, prodorno računalništvo, ambientalna inteligenca, oskrbovana stanovanja	Simulator scenarijev v pametnem prostoru. Simuliramo izkušnjo določene osebe (avatarja) glede na njegove osebne preference in dani kontekst. Avatar se premika po sobah v 3D prostoru po predefinirani poti in aktivira ambientalno logiko glede pogoje, ki smo jih določili. V simulacijo lahko vključimo različne emulirane pa tudi fizične naprave.	V fazi analize zahtev služi za izvedbo različnih uporabniških situacij oz. primerov uporabe v virtualnem pametnem prostoru. Orodje podpira tudi emulacijo naprav in realistično simulacijo komunikacijskega nivoja. Posledično je uporabno za prototipiranje, zasnovo arhitekture sistema kot tudi za finalno testiranje integralne rešitve pred končno namestitvijo na fizične naprave oziroma produkcijsko lokacijo.	aplikativni, komunikacijski, arhitekturni	S priloženim grafičnim vmesnikom modeliramo prostor. Iz menuja izbiramo različne predefinirane objekte - pohištvo in omrežne naprave in jih dodajamo v situacijo. Narišemo pot avatarjev in s tem zasnujemo scenarij. Programske pogoje in pravila definiramo s posebno programsko skripto. Za modeliranje 3D objektov po meri je potrebno uporabiti zunanje 3D grafične urejevalnike.
OMNeT++	[ni omejitev]	Simulacijsko ogrodje, ki omogoča modeliranje in izvedbo simulacij različnih komunikacijskih omrežij. Možne so razširitve z različnimi modeli npr. vrsta medija, poraba energije, propagacija signala ipd. ter različne standardne protokolne sklade, za različne domene. Primerni modeli oz. razširitve za IoT so med drugim INET, MiXiM in Castalia.	Na simuliranem omrežju je možna izvedba določenih osnovnih primerov uporabe. Uporablja se za razvoj protokolov, distribuiranih algoritmov, evaluacijo topologij, arhitekture sistema. V osnovi je to splošni omrežni simulator, za različne domene pa se v praksi uporablja različne razširitve programa. Omogoča priključitev fizičnih naprav, zato se lahko uporablja tudi v fazah testiranja vgradne programske opreme.	komunikacijski, aplikativni, arhitekturni	Najbolje je začeti v knjižnici na spletni strani programa, kjer najdete različne modele in protokolne sklade. Za zasnovo scenarija s pomočjo grafičnega vmesnikom definirajte vozlišča in topologijo (datoteka NED) in parametre delovanja (datoteka ini). Podrobnejšo logiko na vozliščih in postopek simulacije implementirate s skriptnim jezikom. Nekaj prikazov modeliranja in izvedb je možno najti na youtube. Drug način uporabe je da uporabite knjižnic kot modele in jih povežete navzven preko vmesniških vrat in s tem vzpostavljate hibridne simulacije, primer takšne uporabe je Veins.
Veins	povezani promet, ad hoc omrežja vozil, prometna omrežja VANET, mobilna omrežja MANET, V2X, pametna signalizacija	Hibridni simulator, ki povezuje omrežni simulator MiXiM (ta vključuje mobilne in V2X komunikacijske modele, modele naprav, radijskega medija, vplivov okolja) ter simulator SUMO (za generiranje realističnih prometnih scenarijev, generiranje populacije agentov - vozil, potovanja vozil in vizualizacijo prometa). Oba simulatorja tečeta paralelno in sta povezana preko TCP vrat. Premikanje vozil v SUMO se v živo preslikujejo na aktivna vozlišča v OMNeT++, vizualizacijo izvedbe pa spremljamo v obeh programih.	Omogoča izvedbo različnih primerov uporabe na področju povezanega prometa. Omogoča razvoj protokolov in distribuiranih algoritmov. Izvajamo evaluacijo učinkovitosti radijske komunikacije, performans sistema, obremenitev sistema, topologij. Orodje je predvsem uporabno za zgodnjo fazo razvoja.	komunikacijski, aplikativni	Simulacijo parametriramo ločeno v obeh simulatorjih posebej s pomočjo priloženih grafičnih vmesnikov in konfiguracijskih datotek. Za osnovno instalacijo demo simulacije sledite postopku na naslovu http://veins.car2x.org/tutorial/. Priloženi demo primer vsebuje osnovne konfiguracijske datoteke. Za nadaljnje podrobnejše modeliranje uporabite dokumentacijo na naslovu http://www.sumo.dlr.de/userdoc/. Za konfiguracijo OMNeT++ MiXiM scenarija (ini datoteka) pa dokumentacijo na naslovu http://mixim.sourceforge.net/doc/MiXiM/doc/neddoc/index.html. Nekaj predstavitev je možno najti tudi na youtube.
Castalia	WBAN, monitoring pacientov, športnikov, starostnikov, telemedicina	Razširitev platforme OMNeT++, ki vključuje posebne standardne protokole in modele za domeno.	Simulira WBAN. Na tej platformi je možna izvedba osnovnih primerov uporabe za domeno. Omogoča razvoj različnih izvedb podpornih protokolov in algoritmov za domeno. Izvajamo evaluacijo porabe energije, obremenitve resursov, vpliva na zdravje ipd. Uporablja se v zgodnji fazi razvoja aplikacije, zaradi solidnega nabora modelov se v praksi uporablja tudi kot knjižnica za razvoj ločenih simulatorjev.	komunikacijski, aplikativni	Izvedba po principu platforme OMNeT++, referenčni priročnik najdete na naslovu: https://castalia.forge.nicta.com.au/index.php/en/documentation.html
CupCarbon	pametna mesta, mestno omrežje	Simulator IoT v kontekstu mesta. Uporablja OpenStreetMap zemljevide, ki se osvežujejo avtomatsko iz API-ja.	Simulira WSN na določeni lokaciji v mestu. Omogoča izvedbo enostavnejših primerov uporabe za domeno, v kateri nastopajo mobilni agenti (vozila, pešci..) in fiksne omrežne naprave. V simulaciji upošteva pozicije zgradb, kar je uporabno za evaluacijo WSN postavitve na neki lokaciji. Z modelom interferenc lahko preverimo učinkovitost omrežja ob velikem številu vozlišč, npr. ob konicah v mestnem jedru. Možna je tudi evaluacija porabe energije naprav za izvajano aplikacijo in glede na druge vplive okolja. Orodje je uporabno v zgodnejših fazah razvoja.	komunikacijski, aplikativni	Vozlišča pozicionirate neposredno na zemljevid OpenStreetMap. Zemljevid se osvežuje avtomatsko iz online API-jev. Na posameznem vozlišču definirate različne radijske in energetske parametre, mobilno pot, specifično programsko logiko vozlišč implementirate s skriptnim jezikom. Razvojna skupina redno objavlja prezentacije uporabe programa na youtube.
NS-3	[ni omejitev]	Simulator vseh vrst komunikacijskih omrežij (ožičena, brezžična, satelitska), privzeto vključuje različne podporne modele radijske povezave, okolja, porabe energije ter standardne protokolne sklade in možne topologije.	Na simulirani omrežni konfiguraciji je možna izvedba osnovnih primerov uporabe za poljubne IoT domene. Omogoča razvoj komunikacijskih protokolov, distribuiranih algoritmov. Za evaluacijo izbrane sistemske arhitekture, topologij, protokolnih skladov v začetnih fazah razvoja. Omogoča povezovanje na fizične	komunikacijski, aplikativni	Potek simulacije implementirate z jezikom Python, izvedbo na napravah pa v jeziku C++. Začetna navodila najdete na strani https://www.nsnam.org/docs/release/3.25/tutorial/html/index.html

			naprave, zato se lahko uporablja tudi v testnih fazah vgradne programske opreme.		
EASIM	pametne stavbe, učinkovita raba energije, sistemi HVAC, pametno omrežje, prodorni sistemi	Simulator domačega električnega omrežja. Vzpostavlja simulirano okolje sistema električnih porabnikov (naprav), virov električne energije za gospodinjstva (omrežje, solarni paneli, veter) in baterij. Simulator je zgrajen na osnovi C++.	Simulira dinamiko različnih virov, porabnikov in akumulatorjev električne energije v gospodinjstvu. Na tej platformi je možno testirati pravila in strategije za področje učinkovitega upravljanja z električno energijo za pametni dom. S tem podpira razvoj aplikacij za znižanje porabe in stroškov električne energije v gospodinjstvih. Možna uporaba je tudi v aplikacijah za upravljanje pametnega omrežja (smart grid). Generira profile porabnikov - statistiko porabe posameznih gospodinjskih porabnikov, modeliramo stanovanjske soseske. S tem podpira razvoj algoritmov za predikcijo porabe, posledično analitičnih aplikacij in mehanizmov za upravljanje z obremenitvami v električnem omrežju ipd.	aplikativni	Simulator je v fazi prototipa in dokumentacija v angleškem jeziku na žalost še ni na voljo.
SimPy	[ni omejitev]	Tekstualni splošni diskretni simulator, zgrajen po navdihu orodij Simula in Simscript, ki podpira modeliranje scenarijev v jeziku Python. Splošnih diskretnih simulatorjev je sicer veliko, vendar je po našem mnenju jezik Python praktična izbira za modeliranje, zato kot primer izpostavljamo to orodje.	Simulator je uporaben za izvedbo poljubnih primerov uporabe za poljubne domene v IoT. Modeliramo lahko enostavne ali kompleksne scenarije, v katere vključujemo agente, storitve, API-je in druge sisteme, brez da bi se omejevali na podrobnosti kot so tehnologije, karakteristika omrežja, konfiguracija sistema ipd. Uporabni so za evaluacijo scenarijev na konceptualnem nivoju, spoznavanje karakteristik zasnove, ugotavljanje segmentov nedoločenosti itd. S pomočjo priloženih porazdelitvenih funkcij lahko učinkovito modeliramo dinamiko sistema in okolja. Iz tega sledi, da je simulacija na splošnem nivoju uporabna v fazi zajema aplikativnih zahtev IoT sistema, ko tehnologija ni določena, natančnost spremljajočih robnih modelov ni ključna, želimo si predvsem enostavno postavitev modela in dobre performanse za izvedbo simulacije.	aplikativni	Model sistema implementirate v jeziku Python. Izvedba se sproži preko komandne vrstice. Začetna navodila za uporabo se nahajajo na naslovu http://simpy.readthedocs.io/en/latest/simpy_intro/index.html .
Ptolemy II	[ni omejitev]	Splošni diskretni simulator za izgradnjo kompleksnih modelov. V situaciji lahko kombiniramo raznolike stile modeliranja. Na voljo so razširitve za WSN (VisualSense), simulacije zveznega časa (HyVisual). Za platformo je bilo razvito orodje, ki zna prevajati modele v TinyOS aplikacije.	Podpira modeliranje kompleksnih primerov uporabe, ki vključujejo različne podsisteme iz večih različnih domen hkrati. Poleg osnovnih funkcij za diskretne simulacije vsebuje tudi bolj specifičen nabor gradnikov za modeliranje algoritmov iz različnih področij elektrotehnike in računalništva. Posledično se lahko orodje uporablja za hitro prototipiranje vgradnih aplikacij in sistemov realnega časa.	aplikativni (vgradne naprave)	Model implementirate vizuelno s pomočjo relativno močnega grafičnega urejevalnika. Na voljo so različni gradniki, ki jih med seboj povezujete in parametrirate. Uporabite različne stile modeliranja (končni avtomat, podatkovni tok, diskretni dogodki, skriptni jezik). Detajlno izvedbo gradnikov (vozlišč) implementirate v jeziku Java. Začetna navodila za uporabo najdete na naslovu http://ptolemy.eecs.berkeley.edu/ptolemyII/tutorial.htm. Nekaj predstavitev uporabe je možno najti tudi na youtube.
Shawn	sistemi IoT, ki obsegajo veliko število aktivnih senzorjev in aktuatorjev: pametno mesto, povezani promet, spremljanje stanja zgradb (SHM), spremljanje stanja okolja (preprečevanje požarov, poplav), industrijski IoT itd.	Omrežni simulator optimiziran za skalabilna IoT omrežja (>100.000 naprav). To je omogočeno na način, da so modeli komunikacije, protokoli in naprav rešeni približno - algoritemsko.	Simulira veliko število radijsko povezanih naprav v WSN omrežju. Na tej osnovi preverjamo izvajanje specifičnih funkcionalnih delov sistema, distribuiranih algoritmov in protokolov. Vključuje modele izgubnega medija, kar je uporabno za evaluacijo učinkovitosti komunikacije in robustnosti sistema. Omogoča evaluacijo performans in omejitev. S simulatorjem si lahko pomagamo tudi pri evaluaciji arhitekture sistema.	komunikacijski, arhitekturni	Logiko na vozliščih implementiramo v jeziku C++. Simulator je enostavno parametrirati, Simulacija se proži iz komandne vrstice. Vizualizacija izvedbe je mogoča z razširitvijo. Začetna navodila najdete na strani https://github.com/itm/shawn/wiki . V github je veliko primerov simulacij: https://github.com/itm/shawn/tree/master/src/apps .
SIDnet-SWANS	[ni omejitev]	Simulator poljubnih IoT/WSN omrežij. V osnovi je zgrajen iz enote za modeliranje protokolnega sklada (SWANS) ter enote, ki podpira različne modele dinamike okolja (SIDnet). Posebnost je poizvedbeni jezik, s katerim preberemo stanje na vozliščih, ki spominja na SQL. Dobro je zasnovana vizuelna predstavitev porabe energije po vozliščih.	Simulator je uporaben v izvedbenih fazah razvoja IoT sistema. Simulira WSN omrežje in spremljajoče fenomene kot so izpadi, fizikalne spremembe, obremenitve, poraba energije ipd. Na podlagi te dinamike okolja in robnih pogojev je možno testirati IoT aplikacije, distribuirane algoritme in protokole. Velika skalabilnost omogoča evaluacijo omejitev sistema in performans. Simulator je možno povezovati s	komunikacijski, arhitekturni, aplikativni	Logike na vozliščih implementirate v jeziku Java. Zagon simulacije je možen preko komandne vrstice ali Netbeans ali Eclipse IDE. Začetna navodila bost našli na tem naslovu: http://users.eecs.northwestern.edu/~ocg474/SIDnet/SIDnet-SWANS%20manual.pdf

		Nadpovprečna je tudi skalabilnost (> 1.000.000 vozlišč).	fizičnimi napravami, posledično je uporaben tudi v fazah zaključnega testiranja vgradne programske opreme pred namestitvijo v produkcijsko okolje.		
GrooveNet	povezani promet, ad hoc omrežja vozil, prometna omrežja VANET, mobilna omrežja MANET, V2X, sledenje vozilom	Simulator ad hoc omrežij v prometu. Podpira številne modele komunikacije in okolja. Podprta je uporaba zemljevida OpenStreetMap, ki se osvežuje samodejno iz online API-ja. Model mobilnosti je podprt z GPS, posledično je v simulacijo možno vključiti tudi fizična vozila (naprave iz terena). V simulaciji je možno spremljati tudi druge dogodke, ki jih proži sledilna naprava fizičnega vozila, npr. nujna sporočila in opozorila. Na vsakem od vozlič je implementiran lasten model mobilnosti, izvedbe poti in komunikacije. Dodani so tudi modeli spreminjanja pasu, gostote prometa, algoritmi za platooning (car following). Modularna zgradba omogoča razširitve modelov (model varnosti npr.).	Simulira mobilna komunikacijska omrežja in populacije večjega števila vozil (>1000). Na tej osnovi omogoča izvedbo primerov uporabe na področju povezanega prometa. Uporablja se za razvoj protokolov in distribuiranih algoritmov. Izvajamo evaluacijo učinkovitosti radijske komunikacije, performans sistema, obremenitev sistema, topologij. Orodje je uporabno za zgodnjo fazo razvoja.	komunikacijski	Uporaba je enostavna. Podprt je grafični vmesnik za konfiguriranje simulacije. Izberemo priložene modele okolja, komunikacije in druge ter jih parametriramo. S pomočjo avtomatske postavitve množice vozil po območju inicializiramo začetno postavitev. Navodila najdete na tem naslovu: https://github.com/mlab-upenn/GrooveNet. Nekaj predstavitev uporabe je možno najti tudi na youtube.
WorldSens Simulator	[ni omejitev]	Tekstualni simulator WSN omrežja. Sestavljata ga simulator naprav Wsim in simulator okolja WSNet. Naprave so emulirane, podpira različne operacijske sisteme. Vključuje dobro podprt nabor različnih modelov propagacije brezžičnega signala, interferenc, modulacije, anten, modele mobilnosti, porabe energije, fizikalne fenomene.	Zelo natančno simulira oz. emulira vgradne naprave. Simulira različne karakteristike brezžičnega sistema in fenomene okolja. Na tej osnovi je možna evaluacija aplikativne rešitve pred namestitvijo na končne fizične naprave. Ker podpira operacijske sisteme TinyOS, ContikiOS, FreeRTOS, podpira razvoj brez potrebe po fizičnih napravah.	aplikativni	Konfiguracijo WSN in topologijo definirate ročno z XML datoteko. Logiko naprav implementirate v jeziku C. Simulacijo zaženemo preko komandne vrstice. Dokumentacijo najdete na tej strani http://wsim.gforge.inria.fr/tutorial.html
Freedomotic	pametni dom, avtomatizacija doma, oskrbovana stanovanja, pametni prostori, vseprisotno računalništvo, prodorno računalništvo, ambientalna inteligenca	Razvojno okolje / simulator / orodje za prototipiranje za pametne prostore. Simulacija poteka na tlorisu prostora. V scenarij lahko vključujete končne naprave (npr. pametna sijalka, IP kamera), pa tudi poljubne Rest API storitve ipd. Elementi se aktivirajo glede na predefinirane pogoje (dogodke).	Simulira pametni prostor. Podpira izvedbo različnih primerov uporabe v domeni. Knjižnica vsebuje končne naprave, ki jih je enostavno dodajati v situacijo, nastavljati pravila in logično povezovati. Simulator je uporaben v zgodnjih razvojnih fazah - zajema zahtev in konceptualne zasnove. Ker povezuje več tipov naprav na isto platformo ob uporabi standardnih komunikacijskih tehnologij in ponuja možnost vključitve fizičnih naprav, ga lahko uporabljamo tudi kot middleware, za konfiguriranje in daljinsko upravljanje pametnega doma.	aplikativni, arhitekturni	V priloženo grafično orodje uvozimo poljubno tlorisno sliko prostora, ki ga bomo modelirali, označimo stene in sobe. V prostor dodamo naprave, ki so dostopne iz spletne knjižnice ponudnika. Definiramo fizično pot ljudi. Definiramo pravila in pogoje proženja naprav. Izvedemo simulacijo. Začetna navodila najdete na tem naslovu http://freedomotic.com/content/users-tutorial . Nekaj predstavitev je možno najti tudi na youtube.
Avrora	[ni omejitev]	Simulator za mikrokrmilnik Atmel AVR ter periferno WSN platformo MEMSIC Mica2 in MicaZ. Vključene so funkcionalnosti profiler, debugger. Dodan vključen je tudi simulator senzorskega omrežja velikega števila vozlišč.	Simulira skalabilno senzorsko omrežje in ter simulira platformo AVR ter MicaZ naprav. Za natančno evaluacijo vgradne aplikativne rešitve pred namestitvijo na vgradne naprave, podpira debugging, profiling, monitoring rešitve. Omogoča tudi evaluacijo porabe energije. Podpira razvoj na vgradnih napravah brez uporabe fizične naprave.	aplikativni	Program se upravlja preko komandne vrstice. Podamo lahko programe v formatu C, objdump ali assemblerske pakete. Osnovni postopek je opisan tule: http://compilers.cs.ucla.edu/avrora/start.html Postopek za izvedbo simulacije senzorskih omrežij je opisan na tem mestu: http://compilers.cs.ucla.edu/avrora/sensors.html Pomoč je med drugim na voljo tudi preko komandne vrstice.
StreetLightSim	pametna javna razsvetljava, pametno mesto	Simulator razširjen iz Veins (SUMO + OMNeT++), prilagojen za področje pametne razsvetljave. Simulira mobilna komunikacijska omrežja in populacije vozil.	Za izvedbo primerov uporabe na področju pametne javne razsvetljave. Omogoča razvoj in evaluacijo adaptivnih algoritmov, ki upoštevajo različne parametre - hitrost vozil, gostoto prometa, trenutno količino dnevne svetlobe, vremenske razmere, obdobje dneva (ura), dan v tednu, praznik, statistiko nesreč na lokaciji, stopnjo kriminala na lokaciji ipd. Na podlagi teh parametrov je možno opazno zmanjšati operativne stroške na področju pametne razsvetljave, vplive na okolje idr. Orodje je uporabno v vseh fazah razvoja.	aplikativni	Sledite navodilom za instalacijo in zagon demo prezentacij na naslovu http://www.streetlightsim.ecs.soton.ac.uk/
ArduPilot	aplikacije avtonomnih UAV naprav: logistika, kmetijstvo, pametna mesta, okoljevarstvo, sledenje objektom,	Programska oprema za podporo planiranja in izvajanja avtonomnih misij za UAV letalnike, multikopterje, vozila in plovila. Prednost tega pristopa je natančna izvedba misije ter doseg (dlje od območja vidnega polja upravljavca in dosega RC naprave)	Letalniki so okretna robotska senzorska vozlišča, ki delujejo v 3D prostoru. S senzorji zaznavajo različne podatke, z aktuatorji izvajajo akcije. S povezovanjem letalnikov v sodelujoče gruče vzpostavljamo distribuirane senzorske omrežne aplikacije (WSN). Avtonomno misijo si lahko po tej strani predstavljamo kot predprogramiran	aplikativni	S priloženim orodjem načrtujete pot vozila. Začrtano pot odsimulirate na priloženem simulacijskem orodju. Končno pot naložite na kontrolnik vozila. Z RC kontrolnikom aktivirate izvedbo misije. Vozilo avtonomno izvede misijo, zbrane podatke preberemo iz naprave. Na youtube je nekaj

	kartografija, prometna varnost, prodorno računalništvo		postopek - funkcionalnost, orodje pa kot razvojno orodje z vgrajenim simulatorjem in kontrolnimi funkcijami.		predstavitev končne delujoče platforme za različne tipe naprav (multikopter, plovilo).
Mosaik	pametno omrežje (smart grid)	Kosimulator za povezovanje domenskih simulatorjev iz področja pametnega omrežja. Koordinira izmenjavo podatkov v hibridni simulaciji.	Vsebuje različne adapterje za povezovanje domenskih simulatorjev iz področja fotovoltaike (in drugih električnih proizvodnih virov), prenosa električne energije, odjemalcev, pametnih števcev, omrežne komunikacije, orodij za analitiko, logiranje ipd. S tem podpira vzpostavitev hibridnih simulacijskih postavitev za smart grid in omogoča izvedbo kompleksnih primerov uporabe za pametno omrežje.	aplikativni, arhitekturni	Osnovna navodila izvedbe demo simulacije najdete na naslovu https://mosaik.offis.de/docs/ Podprti so različni API-ji za znane prosto dostopne simulatorje in orodja iz področja pametnega omrežja. Scenarije, ki jih izvaja kosimulator, modelirate v jeziku Python.
TRMSim-WSN	[ni omejitev]	Simulator za evaluacijo algoritmov zaupanja in ugleda v senzorskih omrežjih.	Algoritmi zaupanja in ugleda služijo za upravljanje s potencialno nevarnimi vozlišči, jih pravočasno izločajo iz komunikacijske seje.	varnostni (WSN)	Primeri algoritmov zaupanja in ugleda so priloženi programu. Uporaba je enostavna. Na vhodu izberete algoritem in definirate parametre, določite razmerje dobrih in slabih vozlišč ter poženete simulacijo.
Sensor Security Simulator	[ni omejitev]	Simulator za področje raziskovanja varnostnih mehanizmov (algoritmov in protokolov) v WSN.	Program simulira skalabilno brezžično ad hoc senzorsko omrežje. Generator vozlišč in škodljivih vozlišč vzpostavi situacijo, znotraj katere evaluiramo različne varnostne algoritme in protokole, ki so implementirani v platformi.	varnostni (WSN)	Dokumentacija ni na voljo, razen strokovni članki navedeni na strani, podprt pa je grafični vmesnik, ki omogoča parametriranje različnih mehanizmov. Terja določeno predznanje o področju, več o mehanizmih je možno poimensko najti v strokovnih člankih.
WSN Localization Simulator	[ni omejitev]	Simulator za razvoj algoritmov za namen pozicioniranja v prostoru. Za ta namen se uporablja posebne algoritme, ki pa so samo relativno natančni (npr. v izračun so vključena tudi druga vozlišča).	Simulira WSN omrežje. Služi za raziskave in evaluacijo algoritmov za pozicioniranje WSN vozlišč. Na podlagi priloženih modelov mobilnosti, energije in propagacije signala evaluiramo učinkovitost sistema ob uporabi različnih izbranih algoritmov in parametrov delovanja.	komunikacijski, arhitekturni, varnostni	Podprt je grafični vmesnik. Privzeto je v programski paket dodanih več znanih algoritmov. Zgeneriramo polje senzorjev na osnovi podanih parametrov, izberemo lokacijski algoritem in poženemo simulacijo.
Simulatorji konteksta (modeliranje agentov)					
Netlogo	javni prostori, pametna mesta, industrija, kmetijstvo	Orodje za agentno modeliranje. V model je možno vpeljati množice različnih agentov (ljudje, roboti, avtomobili, živali, rastline, bakterije, virusi ipd.) v odnosu do poljubnih zunanjih dejavnikov oz. pogojev, ki jih predefinirate (resursov, omejitev) ter drugih agentov.	Na simulaciji množice agentov lahko evaluiramo naprednejše algoritme iz področja umetne inteligence npr. adaptivne algoritme in aplikacije. Orodje je uporabno tako za modeliranje enostavnejše omrežne simulacije (npr. WSN topologij) kot tudi za simulator konteksta.	aplikativni, kontekst, komunikacijski	Grafični vmesnik lahko na enostaven način prilagodite domeni, ki jo simulirate in model postavite v smiseln kontekst za področje obravnave. Postopek simulacije in notranjo logiko implementirate v jeziku Netlogo. Najbolje je začeti v knjižnici modelov https://ccl.northwestern.edu/netlogo/models/index.cgi Preverite tudi podprte razširitve https://github.com/NetLogo/NetLogo/wiki/Extensions
Siafu	pametni prostori, pametno mesto, industrija, kmetijstvo	Simulator agentov. Vključuje modele mesta in pisarne - priloženi so modeli agentov ljudi in avtomobilov.	Agenti generirajo geolokacijske podatke v standardnem formatu. Simulator lahko priklopimo na drug simulator (npr. omrežni). Siafu tako služi kot real-time generator lokacij agentov ter za prikaz izvedbe (vizualizacijo situacije). Generirane podatke je možno uporabiti tudi za algoritme za strojno učenje. Priloženi so scenariji za pametna mesta in pametni prostor (pisarna). Platforma je odprta in je možno ustvariti poljubni model konteksta.	kontekst	Priloženi so demo modeli (pametno mesto - mesta Leimen, Glasgow, Valencia, pametni prostor - pisarna), ki so dobra osnova za preizkušanje. Navodila za prilagoditve oz. kreiranje modela najdete na tem naslovu: http://siafusimulator.org/tutorial/1.html Za naprednejšo uporabo so navodila na tem mestu http://siafusimulator.org/developers/
SUMO	povezani promet, pametni promet, ad hoc omrežja vozil, prometna omrežja VANET, mobilna omrežja MANET, V2X, sledenje vozilom, pametno mesto	Simulator prometa. Simulira kontekst kompleksnih cestnih omrežij - generira vozila, pešce, signalizacijo. Upošteva fizično infrastrukturo na določeni lokaciji (ceste, pločnike, stavbe, smer prometa). Podatke o kontekstu je možno uvoziti iz OpenStreetMap.	Uporaben je za generiranje realističnih scenarijev v prometu, vizualizacijo simulacije. Simulator generira agente, ki vsebujejo geolokacijsko informacijo. Simulator lahko priklopimo na drug simulator (npr. omrežni ali splošni), uporaben je tudi za generiranje podatkov vzorcev..	kontekst	Za začetna navodila sledite povezavi http://sumo.dlr.de/wiki/Tutorials/Hello_Sumo Nekaj predstavitev je možno najti tudi na youtube.
Bonnmotion	mobilne domene - pametno mesto, povezani promet, pametna pisarna	Vsebuje znane modele mobilnosti (Random walk itd.), na osnovi katerih kreira scenarij.	Scenarije lahko izvozite v ustrezen format in uporabite na vhodu drugih simulatorjev (ns-3, Cooja, MiXiM)	kontekst	Scenarije kreirate preko komandne vrstice. Dokumentacija je dosegljiva na povezavi http://sys.cs.uos.de/bonnmotion/doc/README.pdf
Virtualizatorji					

DPWSim	pametni prostori, industrijska avtomatika, pametni promet	Virtualizator naprav DPWS (Devices Profile for Web Services), t.j. naprav z web service vmesnikom. S tem orodjem vzpostavite virtualizirane naprave v lokalnem omrežju, dosegljive preko IP naslova. Videti so kot dejanske naprave z naborom funkcionalnosti.	Omogoča prototipiranje, razvoj, testiranje distribuiranih aplikacij v SOA arhitekturi brez potrebe po fizičnih napravah.	komunikacijski, arhitekturni	Uporaba je enostavna. Podprt je grafični vmesnik, preko katerega definirate naprave, definirate operacije in dogodke, naprava je dosegljiva preko standarda DPWS.
Hue emulator	pametni prostori	Virtualizator pametne sijalke Philips. API sijalke je dosegljiv na IP omrežju z enakim naborom funkcionalnosti kot fizična naprava. Trenutna funkcija sijalke je grafično prikazana.	Za namen razvoja IoT brez potrebe po uporabe fizične naprave.	aplikativni	Uporaba je enostavna. Navodila so na strani http://steveyo.github.io/Hue-Emulator/
AllJoyn Device Simulator	pametni prostori	Virtualizator naprav AllJoyn. AllJoyn je standard za interoperabilnost v IoT.	S tem orodjem lahko vzpostavite simulirane instance AllJoyn naprav za namen razvoja in prototipiranja IoT aplikacij brez potrebe po fizičnih napravah. Podprta je generična pametna sijalka.	aplikativni	
Simulatorji senzorjev					
SensorSimulator	[ni omejitev]	SensorSimulator je simulator/virtualizatorjev senzorjev kot so senzorji pospeška, svetlobe, magnetnega polja, tlaka, oddaljenosti od predmetov, rotacije, temperature, bar kode, žiroskop, kompas	Senzor je viden v IP omrežju in na voljo za povezovanje v hibridne platforme (SIL simulacije), za podporo razvoja in prototipiranja sistema brez uporabe fizičnih naprav, lahko tudi kot knjižnica za izgradnjo ločene programske simulacijske platforme.	kontekst	Sledite navodilom za instalacijo in zagon demo izvedbe na naslovu https://github.com/openintents/sensorsimulator
tsdbwriter	[ni omejitev]	Simulator oz. generator senzorskih podatkov senzorjev za temperaturo, pretok, hitrost vetra, pritisk ter jih shrani v časovno podatkovno bazo (TSDB). Vrednosti so naključne znotraj vnesenega območja.	Sintetične podatkovne vzorce lahko uporabite v drugih simulatorjih in prototipnih orodjih, stimuluse pri evaluacij IoT aplikacij.	kontekst	Konfiguracija je podprta v JSON formatu. https://dzone.com/articles/c-tool-simulate-iot-sensor
Simulatorji, emulatorji in virtualizatorji vgradnih naprav					
ATEMU	[ni omejitev]	Emulator za mikrokrmilnik Atmel AVR in simulator za periferno WSN platformo MEMSIC Mica2 ter omrežni simulator.	Na emuliranih vozliščih je podprta je izvedba TinyOS aplikacij. Za prototipiranje in razvoj IoT vgradnih programov brez potrebe po fizičnih napravah.	aplikativni (vgradne naprave, WSN)	Dokumentacija posebej ni na voljo, pač pa strokovni članki. Program se upravlja preko komandne vrstice.
SimulAVR	[ni omejitev]	Simulator za družino mikrokrmilnikov Atmel AVR. Napisan je v C++. V osnovi gre za ogrodje, ki je poljubno razširljivo. Posebnega grafičnega vmesnika ni na voljo. Podprta je skriptna izvedba simulacije na podlagi Python skript.	Na simulaciji mikrokrmilnika lahko izvajamo vgradno programsko opremo. Uporablja se za prototipiranje in razvoj brez potrebe po fizični napravi. Podprt je debugger (gdb).	aplikativni (vgradne naprave, WSN)	Program se upravlja preko komandne vrstice. Dokumentacija se nahaja na tem mestu http://www.nongnu.org/simulavr/contents.html
SimAVR	[ni omejitev]	Simulator za družino mikrokrmilnikov Atmel AVR. Platformo je možno enostavno programsko razširjati.	Uporablja se za prototipiranje in razvoj programske opreme za vgradne naprave brez potrebe po fizični napravi. Vključuje funkcionalnost za podrobnejšo analizo poteka programa. Podprt je debugger (gdb).	aplikativni (vgradne naprave, WSN)	Simulator se upravlja preko komandne vrstice. Najbolje je začeti s primeri in dokumentacijo na strani https://github.com/buserror/simavr
Emulare	[ni omejitev]	V osnovi emulator splošne strojne opreme, trenutno je podprta emulacija AVR ATMega, na katerega je možno priključiti različne periferne naprave ter Arduino. Podprt je grafični urejevalnik.	Uporablja se za prototipiranje in razvoj programske opreme za vgradne naprave brez potrebe po fizični napravi. Vključuje funkcionalnosti za podrobnejšo analizo poteka programa. Podprt je debugger (gdb).	aplikativni (vgradne naprave, WSN)	Podprt je grafični urejevalnik. Najbolje je začeti na tej strani: http://emulare.sourceforge.net/
Arduino Debugger/Simulator	[ni omejitev]	Simulator za module Arduino. Vključen je intuitivni grafični vmesnik. Postopek simulacije ni ločen od programa ampak se prevede skupaj z njim (potrebno je Dev-C++ okolje).	Omogoča razhroščevanje Arduino sketch-programov. Uporablja se za podporo razvoju programske opreme za vgradne naprave brez potrebe po fizični napravi.	aplikativni (vgradne naprave, WSN)	Sledite navodilom na strani https://github.com/Paulware/ArduinoDebugger/
Simuino	[ni omejitev]	Simulator za module Arduino UNO in MEGA.	Omogoča razhroščevanje Arduino sketch-programov. Uporablja se za podporo razvoju programske opreme za vgradne naprave brez potrebe po fizični napravi.	aplikativni (vgradne naprave, WSN)	Spletna različica trenutno ni na voljo. Začnite na tej strani http://web.simuino.com/get-started
UnoArduSim	[ni omejitev]	Simulator za Arduino module. Dodanih je več modelov I/O naprav, ki jih lahko vključujemo v izvedbo, npr. različni tipi motorčkov, zvočniki, LED diode in druge analogne naprave... Koda žal ni na voljo, vendar je program brezplačen. Avtor program pogosto pogosto nadgrajuje. Dokumentacija je dobro podprta.	Omogoča razhroščevanje Arduino sketch-programov. Uporablja se za podporo razvoju programske opreme za vgradne naprave brez potrebe po fizični napravi.	aplikativni (vgradne naprave, WSN)	Podprt je grafični urejevalnik. Dokumentacija je priložena programskemu paketu.

EMUL8	[ni omejitev]	Splošni emulator različnih procesorjev, ki med drugim podpira ARM Cortex-A in Cortex-M.	Omogoča razhroščevanje programa za vgradno napravo. Uporablja se za podporo razvoju programske opreme za vgradne naprave brez potrebe po fizični napravi. Vključuje funkcionalnosti za podrobnejšo analizo poteka programa. Podprto je razhroščevanje z gdb.	aplikativni (vgradne naprave, WSN)	Sledite navodilom na tej strani http://emul8.org/get-started
Simulatorji omrežnih algoritmov					
Atarraya	[ni omejitev]	Simulator WSN za razvoj algoritmov za kontrolo topologij. Dodani so enostavni modeli porabe energije in mobilnosti. Programski paket vključuje predloge takšnih algoritmov.	Z ustrezno izbiro algoritmov za kontrolo topologij lahko na nivoju senzorskega omrežja pomembno zmanjšamo porabo energije in s tem razpoložljivost sistema ob zadostni meri povezljivosti. Simulator podpira razvoj in evaluacijo takšnih algoritmov.	komunikacijski (WSN)	Modeliranje je podprto z grafičnim vmesnikom. Izberete algoritem topologije, modele mobilnosti, porabe energije in komunikacije ter druge parametre vezane na izvedbo. Izvajanje simulacije je podprto s solidno vizualizacijo.
Sinalgo	[ni omejitev]	Omrežni simulator za razvoj omrežnih algoritmov. Podpira tudi modeliranje v 3D prostoru. Simulira WSN - mobilnost, komunikacijo, prenos, distribucijo (postavitve vozlišč), interferenco ter izgubni prenosni kanal.	Omogoča hitro prototipiranje, testiranje in evaluacijo različnih omrežnih algoritmov.	komunikacijski (WSN)	Konfiguriranje je podprto z grafičnim vmesnikom in XML konfiguracijskimi skriptami, logiko na vozliščih pa implementirate v jeziku Java. Omogočena je grafična vizualizacija izvedbe v živo. Sledite navodilom na tej strani http://disco.ethz.ch/projects/sinalgo/tutorial/Documentation.html
Operacijski sistemi					
Contiki / Cooja	WSN, protokoli, komunikacija, vgradne naprave	Contiki je operacijski sistem za naprave z omejenimi viri (ARM, AVR, TI SensorTag). Cooja je podporni omrežni simulator / emulator za izvedbo Contiki aplikacij na PC. Simulira WSN in povezane naprave. Vključuje natančne modele komunikacijskih protokolov IPv4, IPv6, 6LowPAN, RPL, CoAP, MQTT. Dodan je model porabe energije.	Na simulirani konfiguraciji je možna natančna izvedba primerov uporabe iz področja WSN in distribuiranih sistemov s poudarkom na natančni izvedbi na aplikativni in komunikacijski ravni. Uporabno je za faze prototipiranja, evaluacijo arhitekture sistema, testiranje vgradne aplikativne rešitve pred namestitvijo na končne fizične naprave, evaluacijo porabo energije, podpira razvoj brez potrebe po fizičnih napravah. Ni namenjen za evaluacijo rešitve na večjem številu vozlišč.	komunikacijski (WSN), aplikativni (vgradne naprave)	Simulacijsko orodje Cooja je zasnovano je na konceptu klasičnih WSN simulatorjev. Podprt je grafični urejevalnik s katerim dodajamo in definiramo vozlišča na ravnini. Na nivoju vozlišča lahko izberete strojno platformo in definirate programsko logiko v jeziku C. Programi na vozliščih so identični tistim, ki tečejo na končnih napravah (pod Contiki OS, obstajajo tudi razširitve za RIOT OS). Krovni scenarij modeliramo s skriptnim jezikom. Seznam podprtih IoT naprav je osvežen na spletni strani proizvajalca. Orodje se pogosto uporablja za raziskovalne in edukativne namene, zato je na voljo relativno veliko je prezentacij na youtube.
TinyOS / TOSSIM	[ni omejitev]	TinyOS je operacijski sistem za vgradne naprave. TOSSIM je simulator za izvedbo programske opreme za TinyOS, napisane v jeziku C. Vzpostavlja virtualno testno okolje množice vgradnih naprav z omejenimi viri povezanih v WSN omrežje. Podpira strojno platformo MICAz.	Za evaluacijo vgradne aplikativne rešitve pred namestitvijo na končne fizične naprave, podpira pa tudi razvoj za TinyOS brez potrebe po fizičnih napravah.	aplikativni (vgradne naprave, WSN), komunikacijski	Programsko kodo naprav implementirate v jeziku C, potek simulacije pa v C++ ali Python. Konfiguracijo omrežne topologije se vzpostavi s tekstualno datoteko. Vsi parametri so podani tekstualno oz. preko komandne vrstice. Zahteva poznavanje Linux, TinyOS. Nekaj predstavitev uporabe je možno najti tudi na youtube.
Razvojna in prototipna orodja					
Reactive Blocks	[ni omejitev]	Platforma za podporo hitrem razvoju in prototipiranju distribuiranih aplikacij za Java. Java je podprta npr. na medmrežnih vmesnikih (npr. Raspberry Pi), tudi usmerjevalnikih.	Platforma podpira izgradnjo aplikacij na nivoju medmrežnih vmesnikov znotraj IoT arhitekture. Omogoča prototipiranje programske opreme. Omogoča razvoj in testiranje, razhroščevanje kot tudi instalacijo na napravo. Orodje je uporabno za vse faze razvoja IoT sistema.	aplikativni (medmrežni vmesniki)	Podprt je grafični vmesnik, ki spominja na UML. V model postavljamo gradne elemente (building blocks), ki jih povezujemo med seboj. Vsak element je točno določenega tipa in vsebuje točno določen vmesnik za povezovanje (koncept API) ter nabor parametrov. Interna logika na elementih se implementira v jeziku Java. Razvijalec lahko uporabi tudi že zgrajene vzorce kode, ki jih objavlja skupnost na strani proizvajalca. Dokumentacija je dobro podprta http://reference.bitreactive.com/ Prezentacije se nahajajo na strani https://www.youtube.com/user/Bitreactive
CodeBlocks Arduino IDE	[ni omejitev]	Integrirano razvojno okolje (IDE) za module Arduino. Gre za razširitev generičnega razvojnega okolja Code::Blocks IDE s funkcionalnostmi za podporo razvoja na Arduino platformi.	Podpira razvojne faze kodiranja (C++) do nameščanja na Arduino napravo. Podprt je tudi simulator, ki je še v začetni razvojni fazi.	aplikativni	Navodila za instalacijo in uporabo najdete na tem mestu https://github.com/provideyourown/CodeBlocks-Arduino

Druga orodja za podporo simulacijam					
wotio/ripple	[ni omejitev]	Generator MQ sporočil. Možno je nastaviti frekvenco oz. bolj podrobne vzorce porajanja sporočil v času. Podprto je nastavljanje topologije.	Orodje vkomponiramo v testno platformo, posebej za testiranje obremenitvenih zmogljivosti sistema.	aplikativni, arhitekturni, komunikacijski, varnostni	Dokumentacije ni na voljo, še najbolj uporaben vir pa je: http://www.interdigital.com/post/iot-architecture-simulation-with-wot-ios-ripple
Automate	[ni omejitev]	Aplikacija za avtomatizacijo opravil v sistemu Android. Glede na stanje določenega dogodka, npr. senzorja (žiroskop, gps) ali kretnje ipd. avtomatsko sprožimo akcije kot je aktivacija kamere, snemanje zvoka, klic REST, sinteza govora itd. Na osnovi določenih pogojev (senzorji in dogodki) torej prožimo poljubne akcije (aktuatorje pametnega telefona). Omogočeno je upravljanje tudi z drugimi aplikacijami preko vključkov kompatibilnih s Tasker ali Locale, npr. povezava z vremenskimi dogodki.	Omogoča hitro prototipiranje mobilnih ali distribuiranih aplikacij na pametnih telefonih, npr. geofencing, brez potrebe po ločenem razvojnem okolju, prevajanju programa. Uporabno bi bilo tudi za implementacijo pomožne distribuirane aplikacije, npr. za nivo medmrežnega vmesnika. Primeri uporabe: pošlji SMS, ko se telefon poveže v točno določeno GSM celico; pošlji e-pošto, ko se bluetooth poveže na točno določeno napravo, kliči funkcijo določene druge naprave, če se telefon obrne na glavo; izklopi zvok, če zazna določeno NFC napravo. Prižgi HI-FI, ko se telefon pojavi v domačen WI-FI omrežju, prižgi snemalnik zvoka na predefiniranem GPS območju.	aplikativni	Koncept je podoben končnemu avtomatu / diagramu stanj. V model dodajamo gradne elemente, ki jih izbiramo iz priložene knjižnice. Elemente parametriramo in jih smiselno povezujemo. Podprta je tudi online platforma za izmenjavo končnih izdelkov med uporabniki. Brezplačna različica ima določene omejitve (max. število gradnikov = 20).
.NET Gadgeteer	[ni omejitev]	Microsoftova platforma za hitro prototipiranje vgradnih naprav. Vključuje posebne standardne kompatibilne module, ki jih je možno enostavno povezati in programirati neposredno v .NET.	Platforma omogoča hiter razvoj prototipov v RAD okolju na ogrodju .NET, po produktivnosti se zaradi tega bolj približa razvoju programske opreme za PC. Za evaluacijo funkcionalnosti kot dopolnitev ali alternativa simulaciji ali kot razvojno orodje za končno platformo .NET Gadgeteer.	aplikativni	Povezovanje modulov v izvedbo je enostavno. Nekaj predstavitev delovanja je možno najti tudi na youtube.

3.2.2 Tabela II - Primerjava po parametrih

Dodatno bomo izvedli primerjavo simulacijskih orodij po določenih praktičnih parametrih (zaloga vrednosti je navedena *kurzivno*)*:*

- Naziv orodja;

UPORABNOST

- Grafični urejevalnik - ali je podprt grafični vmesnik za modeliranje in parametriranje simulacije: *D-Da, N-Ne.*
- Podpora za končne naprave - ali orodje podpira kakšen specifičen tip modula oz. naprave (lahko kot emulacija ali virtualizacija ali kompatibilnost s konkretno strojno platformo): *D-Da, N-Ne.*
- Namestitev na napravo - ali je podprta namestitev programske kode, ki smo jo razvili z modelom, na fizično napravo: *D-Da, N-Ne, e-z nadgradnjo (npr. je na voljo vtičnik).*
- Uvoz zemljevidov (samo za pametna mesta) - ali orodje podpira uvoz geolokacijsko podprtih zemljevidov za kontekst simulacije: *D-Da, N-Ne, DD- zemljevid se osvežuje iz online storitve (API).*

  Ustrezen izris konteksta mesta olajša modeliranje in omogoči atraktivnejšo izvedbo (kar sicer pokriva parameter »Vizualizacija izvedbe« opisan v nadaljevanju).
- Skalabilnost - število aktivnih vozlišč v simulaciji: I - nizko (do 1000), II - srednje (do 100000), III - (več kot 100000)

  Skalabilnost je pomembna za modeliranje IoT sistemov, kjer pričakujemo veliko število hkratno aktivnih naprav v omrežju, npr. ob konicah v mestnem jedru. To je eden ključnih izzivov IoT. Nekateri WSN simulatorji so posebej optimizirani za

evaluacijo skalabilnih postavitev, na račun poenostavitve nekaterih modelov (npr. komunikacijskih).

- Intuitivno za uporabo: ali je orodje intuitivno zasnovano, preprosto za uporabo spoznavanje z orodjem poteka hitro: *D-Da, N-Ne*

SIMULACIJA

- Vizualizacija izvedbe - ali orodje podpira grafično vizualizacijo simulirane situacije v živo (npr. stanje vozlišč, pozicija, raven energije): *D-Da, N-Ne, 3D - podpira 3D vizualizacijo.*

  Vizuelni prikaz izvedbe omogoča boljšo in bolj atraktivno izkušnjo. Kot smo navedli v poglavju 1, splošna orodja zaradi svoje generičnosti niso omejena na kontekst, zato vizualizacije za ta orodja praviloma niso podprte.

- Logiranje izvedbe - ali je podprto beleženje tekočih podatkov o poteku simulacije npr. na zunanjo datoteko: *D-Da, N-Ne.*

- Porazdelitvene funkcije - ali simulacijsko orodje v določenem segmentu podpira modeliranje z verjetnostnimi porazdelitvenimi (npr. normalna, Poissonova itd.): *D-Da, N-Ne.*

- Spreminjanje hitrosti - ali simulator podpira interaktivno spreminjanje hitrosti izvedbe v živo: *D-Da, N-Ne.*

- Interaktivnost - ali je možno spreminjati parametre simulacije v živo med izvedbo: *D-Da, N-Ne.*

- Jezik za izvedbo: programski jezik za kontrolo poteka situacije (simulacijska skripta, batch): *navedba konkretnega programskega jezika, Xml - xml konfiguracijska skripta, namenski - specialen skriptni jezik praviloma posebej razvit za potrebe dane platforme.*

  Ta vidik je pomemben, ker skrajša čas za spoznavanje z orodjem in uporabo simulacije, specialni jeziki pa posebej za nivo simulacij naprav niso ugodni, saj jih je treba še prevajati v jezik fizičnih naprav.

ARHITEKTURA

- Povezljivost - ali je podprta povezljivost simulatorja z drugimi simulatorji preko API, TCP vrat in drugih vmesnikov: *D-Da, N-Ne.*

  Ta karakteristika omogoča povezovanje z drugimi simulatorji in gradnjo hibridnih simulacijskih platform, ter tudi vključevanje fizičnih naprav ali drugih servisov v simulacijo.

- Razširljivost - ali je arhitektura postavljena tako, da omogoča enostavne dograditve in razširitve, npr. je vpeljana solidna objektna in modularna arhitektura, arhitektura vtičnikov: *D-Da, N-Ne.*

MODELI

- Model mobilnosti - ali je podprta mobilnost vozlišč: *GPS - podprto z GPS podatki, D - podprto na drugačen način (npr. s koordinatami X in Y, N-Ne), e - podprto z nadgradnjo.*

  Z mobilnimi vozlišči in geolokacijo lahko modeliramo različne realne primere uporabe npr. s področja prometa in logistike.

- Energetski model - ali je podprt model porabe energije na vozliščih: *D-Da, N-Ne, e-z nadgradnjo.*

  Ta kriterij je predvsem pomemben pri evaluaciji aplikacij za naprave z omejenimi viri, npr. WBAN in WSN.

- Model okolja - ali so podprti modeli fizikalnih vplivov z okolja (npr. temperatura, svetloba ipd.): *D-Da, N-Ne, e-z nadgradnjo.*

  Ta kriterij je pomemben pri evaluaciji primerov uporabe na aplikacijskem nivoju, kjer nastopajo točno določene vrste senzorjev.

- Radijski model - ali je podprta analiza fizičnih lastnosti medija - moč signala, izbira kanala, interference, propagacija signala v primeru ovir ipd. *D-Da, N-Ne, e-z nadgradnjo.*

PROJEKT

- Posodobljeno v 12 mes. - ali je bil program v zadnjih 12 mesecih posodobljen: *D-Da, N-Ne.*
- Skupnost aktivna - ali je orodje živo in podprto s strani skupnosti aktivnih uporabnikov (forum, usenet, IRC, socialna omrežja ipd.): *D-Da, N-Ne.*
- Podprta dokumentacija - ali je dokumentacija zadostno podprta, z navodili za nivo razvoja: *D-Da, N-Ne.*

Opomba: Izpolnjena so tista polja, ki so relevantna za ta tip orodja, informacija je bila dostopna in zanesljiva, sicer so polja neizpolnjena.

Tabela II - primerjava po kriterijih

Naziv	UPORABNOST						SIMULACIJA						ARHIT.		MODELI				PROJEKT		
	Grafični urejevalnik	Podpora za končne naprave	Namestitev na napravo	Uvoz zemljevidov	Skalabilnost (rang)	Intuitivno za uporabo	Vizualizacija izvedbe	Logiranje izvedbe	Porazdelitvene funkcije	Spreminjanje hitrosti	Interaktivnost	Jezik za izvedbo	Povezljivost	Razširljivost	Model mobilnosti	Energetski model	Model okolja	Radijski model	Posodobljeno v 12 mes.	Skupnost aktivna	Podprta dokumentacija
Simulatorji																					
UbiREAL	D						3D	D				CADEL	D				D				
OMNeT++	D				II		D	D	D			namenski, C++	D	D	e	e		e	D	D	D
Veins	D			D			D	D	D			namenski, C++	D	D	GPS			D	D	D	D
Castalia							D		D			namenski, C++			e	D		D		D	
CupCarbon	D			DD	I		D					Senscript			GPS	D		D	D		
NS-3	N				II		D		D			Python, C++	D	D	D			D	D	D	D
EASIM												C				D					
SimPy	N								D	D		Python	D								D
Ptolemy II	D		e			D			D		D	Java				e	e	e	D	D	D
Shawn					III		e		D			C++	D		D		D	D			
SIDnet-SWANS	D				III		D	D		D		Java, namenski	D	D	GPS	D	D	D			D
GrooveNet	D			DD	II	D	D	D				Java	D		GPS						D
WorldSens Simulator	N	D			I			D				C, XML	D			D	D	D			
Freedomotic	D	D				D	D	D	N				D	D	D		N				
Avrora	N	D			II		D	D		D	e	assembler, C		D	e	D		D			D
StreetLightSim																D					
ArduPilot	D	D					e					Python	D		GPS		e		D	D	D
Mosaik												Python	D								D
TRMSim-WSN	D				I	D	D									D		D			
Sensor Security Simulator	D				III	D		D						D							
WSN Localization Simulator	D				I	D	D					C#		D	D	D		D			D
Simulatorji konteksta (modeliranje agentov)																					
Netlogo	D				I	D	D			D		namenski (Netlogo)		D							D
Siafu	D						D					Java	D	D							D
SUMO	D			e	II		D	D				XML	D		GPS						D
Bonnmotion								N							D				D		D
Virtualizatorji																					
DPWSim	D																				
Hue emulator		D			I	D	D					Java	D						D		
AllJoyn Device Simulator	D	D			I	D	D						D						D		N
Simulatorji senzorjev																					
SensorSimulator	N												D				D				
tsdbwriter																	D				N
Simulatorji, emulatorji in virtualizatorji vgradnih naprav																					
ATEMU		D																D	N		N
SimulAVR	N	D										Python								D	D
SimAVR		D					D							D					D		D
Emulare	D	D																			
Arduino Debugger/Simulator	D	D				D	D					Arduino Sketch									N
Simuino	N	D						D				Arduino Sketch									N
UnoArduSim	D	D					D					Arduino Sketch							D		D
EMUL8	N	D										namenski		D					D		D

Simulatorji omrežnih algoritmov																				
Atarraya	D				I	D	D							D	D		D			
Sinalgo	D				III		D	D	D			Java			D			D			
Operacijski sistemi																				
Contiki / Cooja	D	D	e		I			D				namenski		D	N	D				
TinyOS / TOSSIM	N	D			II			D				C++, Python				e	N	D	N	D	D
Razvojna in prototipna orodja																				
Reactive Blocks	D		D			D						Java	D	D					D	D	D
CodeBlocks Arduino IDE	D	D	D			D						C++							D	D	D
Druga orodja za podporo simulacijam																				
wotio/ripple													D							
Automate	D	D				D		D						D					D	D	
.NET Gadgeter	D				I							.NET									

4 Primer uporabe

S senzoriko in IoT medicinskimi napravami bi želeli opremiti stanovanje - oskrbovano stanovanje. Spremljali bi oskrbovanca bolnika z diabetesom. Zajemali bi razne medicinske podatke o oskrbovancu (krvni tlak, telesna temperatura, telesna teža, pulz, stopnja glukoze v krvi ipd.). Poleg tega bi spremljali še druge podatke, ki bodo v pomoč pri interpretaciji zajetih medicinskih podatkov. Če bolniku merimo stopnjo sladkorja v krvi, je pomembno, ali si jo je meril pred ali po obroku, ali če merimo pulz, ni vseeno, ali je oskrbovanec prej bil aktiven ali ne. Prek senzorjev na vratih, predalih itn. ter senzorjev premikanja lahko ugotavljamo, kdaj je bil oskrbovanec v kuhinji, kdaj v kopalnici, kdaj je vstal, kdaj je jedel, koliko se je gibal itn. Zajeti podatki se pošiljajo v nacionalno zdravstveno platformo, kjer se v realnem času obravnavajo in dajejo zdravniškemu osebju boljšo podlago za odločanje ter ukrepanje, npr. da pravočasno preprečimo zdravstvena obolenja.

Nakup vse senzorike, še posebej pa IoT medicinskih naprav, je drag. Za opisan primer bi radi najprej naredili simulacijo. S kakšnim simulatorjem to narediti? Prikažimo, kako si pri tem pomagati z izsledki tega dela. Postopek izbire simulacijskih orodij smo sicer že nakazali v poglavju 2.5. »Izgradnja ciljnega modela in izbira simulacijskih orodij«. Na tem mestu bomo uporabili še tabelo, ki je opisana v poglavju 3.2 »Primerjalna tabela med zbranimi orodji«. Navedli bomo relevantne lastnosti simulatorjev in mesta v tabeli.

Najprej za dani primer uporabe določimo aplikativno domeno. Primer se uvršča v naslednje domene: pametno zdravstvo, telemedicina, oskrbovana stanovanja, pametni prostori, prodorni sistemi.

V Tabeli I (stolpec *Domena IoT*) lociramo orodja: **UbiREAL**, **Freedomotic** in **Castalia**.

Simulatorja **UbiREAL** in **Freedomotic** vključujeta vizualizacijo konteksta (vir: Tabela II - *Vizualizacija izvedbe*). Oba simulatorja se lahko uporabita za postavitev modela arhitekture sistema (t.j. komponent sistema). UbiREAL omogoča bolj podrobne modele za evaluacijo na komunikacijski ravni (vir: Tabela I - *Nivo sistema*).

Simulatorja v osnovi pokrivata področje pametnega doma, z ustrezno prilagoditvijo bi se lahko uporabljala tudi na področju oskrbovanih stanovanj in telemedicine. Potrebna programska dograditev z modeli naprav - senzorjev, merilcev za področje oskrbe pacientov. Freedomotic v smislu takšne razširljivosti obeta več, saj je vzpostavljena programska arhitektura vtičnikov (Tabela II - *Razširljivost*). Pri izvedbi razširitev je v pomoč tudi dobro podprta dokumentacija (Tabela II - *Podprta dokumentacija*).

Oba simulatorja omogočata tudi povezljivost z drugimi programskimi orodji (Tabela II - *Povezljivost*), npr. preko TCP vrat. Namesto da sami programsko razvijamo modele naprav za telesno omrežje, lahko odvedemo izvedbo tega nivoja na drug simulator. Npr. na izbrano simulacijsko platformo za pametni dom (npr. Freedomotic) povežemo simulator za telesno

WBAN omrežje Castalia (Tabela I - *Domena IoT*), ki prevzame simulacijo senzorskih naprav za merjenje temperature, pulza, tlaka, merjenje sladkorja v krvi (npr. inzulinska črpalka). Castalia vsebuje tudi modele za evaluacijo porabe energije (Tabela II - *Energetski model*), poleg tega pa še model mobilnosti (preko nadgradnje - Tabela II, *Model mobilnosti*) in model propagacije signala (Tabela II - *Radijski model*). Na tej osnovi je v enem orodju možna natančna evaluacija aplikativne rešitve za medicinske naprave.

Različne komponente ali skupine komponent lahko tako pokrivajo različni specializirani simulatorji. Npr. za modeliranje programske logike na nivoju medmrežnega vmesnika (ima več resursov kot senzorska naprava, ga je lažje programirati) lahko uporabimo orodje Reactive Blocks (Tabela II - *Funkcionalni opis*). Načeloma lahko v model povežemo tudi dejanske online storitve. V simulacijo navadno lahko vključimo tudi fizične naprave (vidne v IP omrežju). lahko pa naprave tudi virtualiziramo - npr. z virtualizatorjem za DPWS sklad DPWSim.

Za maksimalno natančno testiranje aplikativne izvedbe na nivoju naprav lahko uporabimo simulatorje vgradnih naprav (Tabela I - *Nivo sistema* - omemba »vgradne naprave«) najdemo vrsto orodij: emulatorje (npr. Avrora, ATEMU, SimulAVR ipd.) in simulatorje (Cooja, TOSSIM, Ptolemy II). Npr. simulator Cooja ima implementiran grafični urejevalnik (Tabela II - *Grafični urejevalnik*) in je funkcionalno podobna drugim omrežnim simulatorjem (vozlišča, topologije), s to razliko, da je vezana na operacijski sistem realnega časa Contiki. Programsko kodo, ki jo ustvarimo s simulacijo, je možno potrem brez sintaktičnega prevajanja uporabiti neposredno na fizičnih napravah.

Za boljši model življenjskih navad oskrbovanca oz. celo več oskrbovancev (npr. gospodinjska skupnost) lahko uporabimo še orodje za modeliranje agentov. Simulirani agent bi generiral ljudi, njihove pozicije in tudi določene aktivnosti v prostoru ter nadomestil model mobilnosti v omrežni simulaciji. Orodje se priključi preko TCP vrat (Tabela II - *Povezljivost*). Simulator te vrste je Siafu. Poleg modeliranja prostora ter aktivnosti oskrbovanca pridobimo tudi grafično animacijo izvedbe (Tabela II - *Vizualizacija izvedbe*).

V kolikor vizualizacija prostora in prevelika natančnost izvedbe nista potrebni, je celoten model distribuiranega sistema najlažje zasnovati v celoti kar v omrežnem simulatorju (npr. Castalia, ali NS-3 ali SIDnet-SWANS). Manjkajoče komponente potem vzpostavimo s približnimi modeli, odvisno od zahtevane natančnosti. Castalia je programska razširitev splošnega simulacijskega ogrodja OMNeT++, zato lahko modeliramo ne samo telesne senzorje, ampak tudi druge že omenjene resurse (senzor števdilnika, senzorji na predalnikih, na vratih), aktuatorji (LCD ekran za opomnik) ter storitve oblaka (nacionalna platforma). Ob izbiri osnovne simulacijske platforme je dobro na spletni strani avtorja pregledati, ali mogoče obstajajo kakšni modeli in razširitve, ki bi bili uporabni za naš primer modeliranja.

Potencialno uporabno orodje za našo domeno je uporaben še generator MQ sporočil wotio/ripple. Na različnih segmentih sistema bi izvajali promet sporočil z veliko hitrostjo in merili stopnjo zasičenja ter s tem izvajali performančno evaluacijo. Za nivo evaluacije varnosti pa ga lahko uporabimo tudi kot simulator DoS napada.

Za natančnejšo simulacijo posameznih delov sistema so uporabni tudi drugi simulatorji, vendar so specializirani za raziskovanje partikularnih segmentov in niso omejeni na domeno (Domena IoT = ni omejitev), npr. WSN Localization Simulator ali Sinalgo ipd. V primerjalnih tabelah v poglavju 3.2 smo navedli samo orodja, ki so vsaj minimalno primerljiva, solidno dokumentirana in opisana oz. jih je bilo možno preveriti. Metodologij, formalizmov, ontologij npr. v tabeli nismo navajali. Vsekakor je pri izbiri ustreznih orodij smiselno preveriti tudi druga orodja, navedena v poglavju 3.1 »Opisi simulacijskih orodij in rešitev «. Za dani primer sta tako še zanimivi vmesno programje DomoNet, formalizem za opis pametnega doma DomoML.

Več informacij o posameznem orodju pa lahko najdete v poglavju 3.1 »Opisi simulacijskih orodij in rešitev «.

5 Zaključki

Osnovno nalogo smo začeli s praktičnim scenarijem IoT, ki je vključeval različne senzorske naprave, storitve in druge vire iz različnih aplikativnih domen (pametni dom, pametno mesto, povezani promet). Kakšen simulator izbrati za simulacijo takšnega IoT scenarija?

Pristopili smo nadvse optimistično - da obstaja en simulator, s katerim bi lahko odsimulirali poljuben IoT scenarij. V simulacijo bi dodajali različne vnaprej pripravljene modele znanih IoT produktov, od naprav za pametni dom do pametne opreme v avtomobilu ter na tej podlagi odsimulirali raznovrstne primere uporabe v različnih grafično podprtih kontekstih.

Kmalu smo spoznali, da en simulator, s katerim bi elegantno podprli poljuben IoT scenarij, ne obstaja. Ne obstaja pa pravzaprav iz preprostega razloga. IoT ni samo posamezna pametna ura oz. nakit, kamera ali temperaturni senzor na vremenski postaji, ampak dinamično povezovanje takšnih naprav (in tudi drugih storitev) v sofisticirano aplikabilno izvedbo. Senzorske naprave na eni strani ustvarjajo podatke, ki se na drugi strani zlivajo na oblak. Na oblaku so na voljo teoretično neomejeni strežniški resursi. Z naprednimi algoritmi se obdelujejo masovni podatki pretočeni iz naprav. Ob različnih pogojih se nato prožijo različni dogodki, npr. kot povratne akcije aktuatorjev na terenu. Izvedba IoT scenarija prepleta resurse oz. podatke večje populacije heterogenih komponent iz različnih vsebinskih domen - senzorjev in aktuatorjev, internetnih virov ter drugih storitev v popolnoma nove uporabniške izkušnje.

Obstoječi senzorski sistemi so vezani na področja M2M, WSN in vgradne naprave, pa tudi domenska področja kot so povezani promet, pametni dom, prodorno računalništvo, vseprisotno računalništvo itd. Z razvojem teh področij so nastala specialna simulacijska orodja, med njimi stabilne in dobro vzdrževane platforme. Ta simulacijska orodja skupaj z razvojem IoT pridobivajo na popularnosti, to je do neke mere tudi vidno npr. na statistiki dolpotegov orodij na github.

S pregledovanjem tega terena se je odprla popolnoma nova raven razumevanja potreb po simulacijah v IoT. Tako kot se raznolike funkcionalnosti in viri IoT ad hoc povezujejo v IoT scenarij, lahko kot lego-kocke povezujemo različna programska orodja, od katerih vsako predstavlja del sistema (komponento sistema, množico komponent, določen tehnološki segment). Preko TCP vrat ali drugih vmesnikov povezujemo simulatorje omrežja, generatorje podatkov, emulatorje naprav in druga orodja v unikaten model IoT sistema. Distribuiranemu sistemu IoT ustreza distribuirana oz. hibridna simulacija. S takšno strategijo lahko vzpostavimo zelo natančen model simulacije kompleksnega IoT sistema.

Za osnovo distribuiranega modela najlažje uporabimo omrežni simulator kot npr. OMNeT++. V situacijo postavimo več senzorskih naprav, vsaka od njih izvaja določeno logiko. Ta orodja se subspecializirajo na WSN simulatorje s podrobnejšimi funkcionalnostmi brezžičnega

omrežja kot je modeliranje porabe energije. Evaluacija porabe energije je pomembna, saj senzorske vgradne naprave praviloma nimajo na voljo stalnega vira energije, kar pomembno vpliva na konfiguracijo sistema in izvedbo, npr. uporabo operacijskih sistemov za naprave z omejenimi viri. V bodoče je pričakovati, da se bodo cene naprav še zniževale, performanse varčnih naprav izboljševale, vgradne naprave bodo lažje za uporabo, mogoče bolj generične, naprave bodo konvergirale v smer bolj zmogljivih računalnikov in gostile višje operacijske sistem (npr. Linux, Android), kar omogoča več funkcionalnosti in tudi poceni razvoj.

Posebej zanimiva je uporaba simulatorja konteksta oz. agentnega modeliranja za simulacijo sistema IoT. Agenti so uporabniki, drugi ljudje, živali, ki se premikajo po prostoru in vplivajo na sistem. Simulator agentov spojimo na simulacijsko postavitev in s tem dosežemo, da gibanje in vedenje agentov realistično »poganja« simulacijo IoT sistema. Za kontekst definiramo še določene zakonitosti, omejitve okolja (npr. ljudje, ki se gibajo po prostorih, množica ljudi na javnem mestu) ter grafično zasnovo. S takšnim kontekstom dosežemo večjo realističnost simulacije, vpeljemo realistične zakone okolja, poleg tega pa pridobimo še grafični kontekst.

V zvezi z izgradnjo hibridnega modela je potrebno omeniti, da je potrebno na nek način rešiti še sinhronizacijo, t.j. usklajevanje simulatorjev. Splošno znana prednost simulatorjev je, da lahko scenarij ponovimo, ustavimo, zavrtimo nazaj, pohitrimo, upočasnimo. V hibridni simulaciji je to težje izvesti, kljub temu določene rešitve obstajajo (npr. simulacija vključuje fizično napravo, npr. ko je na vrsti za izvajanje fizično priklopljena naprava, se izvajanje simulacije preklopi na realni čas in po končanju spet pospeši/upočasni izvedbo). Za takšno usklajevanje in sinhronizacijo skrbijo kosimulatorji.

Če želimo brez zgoraj opisanih omejitev in na enostaven način modelirati kompleksne scenarije, ki obsegajo več domen in podsistemov, večja natančnost pa ni potrebna, so še vedno na voljo tudi splošni simulatorji diskretnih dogodkov. V tem primeru zaobidemo tehnološke podrobnosti, osredotočimo se lahko bolj na samo izvedbo primerov uporabe (aplikativni nivo) in tako na kar najbolj produktiven način predstavimo kompleksnejše situacije. Seveda je v tem primeru modele posameznih členov oz. komponent sistema potrebno šele razviti do zadostne natančnosti. Po analogiji s prej opisano strategijo pa lahko na splošni simulator povežemo tudi specifične simulatorje, npr. simulator senzorjev ali emulator vgradne naprave. Lahko pa splošni simulator nastopa tudi samo kot simulacijska komponenta v hibridni simulaciji.

Kombinacij povezovanja (odprtokodnih) simulacijskih orodij in drugih virov v hibridne modele je neomejeno, podobno kot je neomejeno možnosti realizacije povezanih aplikacij in primerov uporabe v IoT. V model lahko vključujemo celo zunanje vire (REST, spletni servisi). Vključujemo lahko tudi fizične naprave (HIL), kar je v pomoč testiranju v zadnjih sklepnih fazah projekta (na simulirani situaciji izvajamo končno rešitev s končno fizično

napravo). V tem primeru simulacija vzpostavlja virtualno testno okolje. Ni slučaj, da določeni simulatorji omogočajo tudi prevedbo in instalacijo (simulirane) programske kode na končne naprave. V tem primeru se simulator razširja v razvojno platformo.

Opazen je trend razvoja IoT z uporabo prosto dostopnih ali odprtokodnih podpornih tehnologij, bolj kot smo tega vajeni pri drugih IT projektih. Smisel IoT je sodelovanje zelo različnih naprav pri opravilu, ključni predpogoj za to pa je medsebojna povezljivost naprav. Zaprte licenčne rešitve niso prilagodljive, kar je nujen predpogoj za svobodno povezovanje (pod)sistemov. Posledično izgleda, da imajo odprte standardne rešitve za razvoj IoT dolgoročno gledano več potenciala kot licenčne (npr. 6LoWPAN proti ZigBee). Ključen faktor pri tem je seveda enostavnost uporabe, pa tudi obstoječi tržni deleži iz obdobja pred IoT.

Še en razlog za razširjenost odprtokodnih orodij pri razvoju IoT je, da se nahajamo v zgodnejši fazi penetracije tehnologije. Obstaja še veliko kreativnega potenciala, motor razvoja so visoko motivirana garažna podjetja oz. start-upi. Resursi razvoja se prilagajajo velikosti projektov - strošek oblaka se obračunava po porabi, razvoj poteka na prostodostopnih in odprtokodnih orodjih. Takšno »demokratičnost« od spodaj navzgor smo že večkrat videli na področju razvoja programske opreme. IoT je v osnovi razvoj fizičnih naprav široke potrošnje, vendar je ključna pri tem računalniška izkušnja, razvijalci se zavedajo pomena odprtosti. Tudi večji proizvajalci so se jim približali, prilagodili in ponudili odprtokodne ali prostodostopne rešitve.

Ponudbe na področju odprtokodnih simulatorjev za širša področja aplikabilna za IoT je res veliko, saj kot smo omenili, IoT v največji meri rekombinira obstoječe sisteme, to je sistem sistemov. Na internetu smo poiskali prostodostopna orodja, ki so dobro podprta z dokumentacijo, jih je bilo možno preveriti. Vključili smo zanimiva, posebna orodja. Veliko orodij je izpadlo iz končnega seznama, ker so redundantna, nevzdrževana, nedokumentirana ipd. Vsak vir je bilo potrebno preveriti, pregledati različne reference, poskušati ugotoviti, ali določeno orodje ustreza naši obravnavi, v čem je posebno in za nas koristno. Zgodilo se je, da smo s kakšnim orodjem porabili dosti časa in na koncu ugotovili, da ne ustreza. Zasledovali smo cilj, da predelamo res vse internetne vire, članke, prezentacije, da pridobimo temeljit pregled trenutnega stanja z različnimi orodji za različne potrebe simulacije za IoT.

Dobljeni pregled bo v pomoč raziskovalcem in razvojnikom IoT sistemov. Razvijalcem IoT pri izbiri ustreznih orodij za evaluacijo aplikacije, programerjem simulatorjev pri iskanju odprtokodnih platform za ponovno uporabo in razširjanje orodij in knjižnic, sistemskim inženirjem pri iskanju idealne postavitve naprav v prostoru itd. Da bi poiskali ustrezno orodje, je potrebno dobro poznati značilnosti IoT sistema, ki ga modeliramo s primerom uporabe. Primer uporabe IoT razčlenimo vertikalno po nivojih sistema (strojna oprema, aplikativni nivo, komunikacija, varnost) in horizontalno po domeni (pametni dom, pametno mesto,

povezani promet, industrijski IoT itd.). Na ta način identificiramo komponente in segmente sistema. Po podobnih rezih so kategorizirana tudi zajeta orodja. Levji delež obravnavanih orodij smo predstavili s primerjalno tabelo, ki bo v veliko pomoč pri iskanju orodij.

Fizične naprave niso samo drage za nakup, razvoj na njih predvsem ni praktičen. Simulatorji učinkovito virtualizirajo celoten proces razvoja distribuiranega sistema IoT do namestitve v končno okolje. S tem prevzemajo vlogo razvojnih okolij, na tem področju je trenutno veliko razvojnega potenciala. Razvojna okolja za IoT bodo v prihodnje vse bolj podobna RAD okoljem. Razvoj IoT bo vse bolj podoben razvoju klasičnih aplikacij, vgradne naprave (moduli) bodo virtualizirane in emulirane, vključen bo omrežni simulator. Eventuelno bo mogoče vključena celo osnovna CAD funkcionalnost za zasnovo ohišja naprave s podporo za 3D tisk. Širši množici programerjev in zanesenjakov bo omogočeno, da si v pičlem popoldnevu izdelajo svojo lastno fizično IoT napravo za osebno uporabo, podobno kot si sprogramirajo program v višjem programskem jeziku. Vrzel med klasičnimi programerji in razvojniki IoT se bo tako še zmanjševala, ključna hrbtenica razvoja IoT pa bo ostala simulacijska funkcionalnost.

Literatura

[1] „What Is Internet of Things (IoT)? A Webopedia Definition“. [Na spletu]. Dostopno: http://www.webopedia.com/TERM/I/internet_of_things.html.

[2] „What is Sensor and What are Different Types of Sensors - EngineersGarage“. [Na spletu]. Dostopno: http://www.engineersgarage.com/articles/sensors.

[3] „Cyber-physical systems and smart cities“, 20-apr-2015. [Na spletu]. Dostopno: http://www.ibm.com/developerworks/library/ba-cyber-physical-systems-and-smart-cities-iot/index.html.

[4] „Operational Technology (OT)“, *Gartner IT Glossary*, 07-nov-2012. [Na spletu]. Dostopno: http://www.gartner.com/it-glossary/operational-technology-ot/.

[5] „Unlocking the potential of the Internet of Things | McKinsey & Company“. [Na spletu]. Dostopno: http://www.mckinsey.com/insights/business_technology/the_internet_of_things_the_value_of_digitizing_the_physical_world.

[6] „The Internet of Things Show | Iot-Inc : E2: Industrial IoT – Ready for an IP Make-over“. [Na spletu]. Dostopno: http://iotbiz.libsyn.com/e2-industrial-iot-ready-for-an-ip-make-over.

[7] „The Internet of Things Show | Iot-Inc : E3: IPSO – IoT Standardization Challenges“. [Na spletu]. Dostopno: http://iotbiz.libsyn.com/e3-ipso-iot-standardization-challenges.

[8] „Roadmap to Trillion Sensors Forks | EE Times“, *EETimes*. [Na spletu]. Dostopno: http://www.eetimes.com/document.asp?doc_id=1328466.

[9] O. A. Mohammed Abo-Zahhad, „A Survey on Protocols, Platforms and Simulation Tools for Wireless Sensor Networks“, *Int. J. Energy Inf. Commun.*, let. 5, št. 6, str. 17–34, 2014.

[10] „Sensor“, *Wikipedia, the free encyclopedia*. 18-feb-2016.

[11] J. Bruneau, W. Jouve, in C. Consel, „DiaSim: A Parameterized Simulator for Pervasive Computing Applications“, v *6th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous'09)*, Toronto, Canada, 2009.

[12] S. Lavirotte, J.-Y. Tigli, G. Rocher, L. El Beze, in A. Palma, *A Dynamic Visual Simulation Environment for Internet of Things*. 2015.

[13] *The Internet of Things Show | Iot-Inc : 04: Part 1 - Sensors from the Inside Out and the Outside In*. .

[14] D. Reinsel, „Data Veracity is a Must in the IoT World - IDC Community“, 28-apr-2015. [Na spletu]. Dostopno: https://idc-community.com/groups/it_agenda/iot/data_veracity_is_a_must_in_the_iot_world~1.

[15] „CupCarbon: A Multi-Agent and Discrete Event Wireless Sensor Network Design and Simulation Tool | The Insight Centre for Data Analytics“. [Na spletu]. Dostopno: https://www.insight-centre.org/content/cupcarbon-multi-agent-and-discrete-event-wireless-sensor-network-design-and-simulation-tool.

[16] „Gateway (telecommunications)“, *Wikipedia, the free encyclopedia*. 14-jun-2016.

[17] „Bitreactive“. [Na spletu]. Dostopno: http://www.bitreactive.com/.

[18] „Presentation ‚Wireless IO Networking with ADAM-2000 Series Sales webinar Paul Diepstraten Product Sales Manager March, 7 th 2013.‘“ [Na spletu]. Dostopno: http://slideplayer.com/slide/5894795/.

[19] „GSMA Mobile IoT Initiative Welcomes First Low Power Wide Area Solutions at Mobile World Congress“, *Newsroom*. [Na spletu]. Dostopno: http://www.gsma.com/newsroom/press-release/gsma-mobile-iot-initiative-welcomes-first-low-power-wide-area-solutions-at-mwc/.

[20] „ZigBee“, *Wikipedia, the free encyclopedia*. 07-feb-2016.

[21] „Difference between Zigbee and 6LowPAN explained with a simple Arduino + Xbee setup“, *LinkedIn Pulse*, 02-avg-2014. [Na spletu]. Dostopno: https://www.linkedin.com/pulse/20140802224850-81482458-difference-between-zigbee-and-6lowpan-explained-with-a-simple-arduino-xbee-setup.

[22] „Gartner Says 6.4 Billion Connected ‚Things‘ Will Be in Use in 2016, Up 30 Percent From 2015 | Business Wire“. [Na spletu]. Dostopno: http://www.businesswire.com/news/home/20151110006335/en/Gartner-6.4-Billion-Connected-2016-30-Percent.

[23] „What is Next for IoT?“, *OpenMind*, 29-dec-2015. [Na spletu]. Dostopno: https://www.bbvaopenmind.com/en/what-is-next-for-iot/.

[24] „The Internet of Things and the Importance of Modeling and Simulation | Automation World“. [Na spletu]. Dostopno: http://www.automationworld.com/all/internet-things-and-importance-modeling-and-simulation.

[25] „Testing the Internet of Things? It’s time to plan your test strategy“, *IoT Now - How to run an IoT enabled business*, 25-maj-2015. [Na spletu]. Dostopno: http://www.iot-now.com/2015/05/25/33241-testing-the-internet-of-things-its-time-to-plan-your-test-strategy/.

[26] M. Bielert, „HAEC-SIM: A Simulation Framework for Highly Adaptive Energy-Efficient Computing Platforms - SIMUTOOLS 2015 Athens, Greece - P504.PDF“.

[27] J. M. Stecklein, J. Dabney, B. Dick, B. Haskins, R. Lovell, in G. Moroney, „Error Cost Escalation Through the Project Life Cycle“, predstavljeno na 14th Annual International Symposium, Toulouse, France, 2004.

[28] G. Brambilla, A. Grazioli, M. Picone, F. Zanichelli, in M. Amoretti, „A cost-effective approach to software-in-the-loop simulation of pervasive systems and applications“, v *2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2014, str. 207–210.

[29] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, in F. Zanichelli, „A Simulation Platform for Large-Scale Internet of Things Scenarios in Urban Environments“, 2014.

[30] J. Virant, *Modeliranje in simuliranje računalniških sistemov*. Radovljica: Didakta, 1991.

[31] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño, in J. García-Haro, „Simulation Tools for Wireless Sensor Networks“, *ResearchGate*.

[32] „Emul8 lets you develop embedded systems in your PC.“ [Na spletu]. Dostopno: http://emul8.org/learn-more.

[33] M. Kabač, N. Volanschi, in C. Consel, „An Evaluation of the DiaSuite Toolset by Professional Developers“, 2015.

[34] „FIT/IoT-LAB • Very large scale open wireless sensor network testbed“. [Na spletu]. Dostopno: https://www.iot-lab.info/.

[35] E. S. Reetz, Kuemper, Daniel, Moessner, Klaus, in Tönjes, Ralf, „How to Test IoT-based Services before Deploying them into Real World.pdf“, 2013.

[36] Gradinaru, Bogdan, „Software-in-the-loop and hardware-in-the-loop simulations (or simply SILS and HILS)“, *Bazaar 2.0*, 02-mar-2010. [Na spletu]. Dostopno: https://bogdangradinaru.wordpress.com/2010/03/02/software-in-the-loop-and-hardware-in-the-loop-simulations-or-simply-sils-and-hils/.

[37] „Cloud Computing Simulators: A Detailed Survey and Future Direction (PDF Download Available)“. [Na spletu]. Dostopno: https://www.researchgate.net/publication/261436186_Cloud_Computing_Simulators_A_Detailed_Survey_and_Future_Direction.

[38] „UbiREAL: Realistic Smartspace Simulator for Systematic Testing“. [Na spletu]. Dostopno: http://ubireal.org/pdffiles/ubicomp06-nishikawa.pdf.

[39] Varga, Andras, „OMNeT++ Summit 2015 - Presentation #1 by Andras Varga“, 04-nov-2015.

[40] „OMNeT++ Community Summit / International Workshop on OMNeT++“. [Na spletu]. Dostopno: https://summit.omnetpp.org/archive/.

[41] „INET Framework - Model Catalog“. [Na spletu]. Dostopno: https://inet.omnetpp.org/Protocols.html.

[42] „Modules - Veins“. [Na spletu]. Dostopno: http://veins.car2x.org/documentation/modules/.

[43] M. Lounis, K. Medhi, in A. Bounceur, „A CupCarbon Tool for Simulating Destructive Insect Movements“, *ResearchGate*, mar. 2014.

[44] „Project PERSEPTEUR (3D Virtual Platform for Wireless Sensor Network Simulation) | ANR - Agence Nationale de la Recherche“. [Na spletu]. Dostopno: http://www.agence-nationale-recherche.fr/?Project=ANR-14-CE24-0017.

[45] F. Bouakkaz, M. Omar, A. Bounceur, in A. Tari, „Secure and Efficient Sharing Aggregation Scheme for Data Protection in WSNs“, predstavljeno na ISSPIT, 2015.

[46] „What is ns-3 « ns-3“. [Na spletu]. Dostopno: https://www.nsnam.org/overview/what-is-ns-3/.

[47] „ns-3 Model Library — Model Library“. [Na spletu]. Dostopno: https://www.nsnam.org/docs/models/html/index.html.

[48] „iTETRIS Platform“. [Na spletu]. Dostopno: http://www.ict-itetris.eu/itetris_platform.html.

[49] „Itetris simulation Projects“, *Network Simulation Tools*. [Na spletu]. Dostopno: https://networksimulationtools.com/itetris-simulation/.

[50] „iTETRIS FAQ“. [Na spletu]. Dostopno: http://www.ict-itetris.eu/10-10-10-community/faq/.

[51] „COLOMBO Project“. [Na spletu]. Dostopno: http://colombo-fp7.eu/.

[52] „ns (simulator)“, *Wikipedia, the free encyclopedia*. 17-feb-2016.

[53] B. Musznicki in P. Zwierzykowski, „Survey of Simulators for Wireless Sensor Networks“, *ResearchGate*, let. 5, št. 3, str. 23–50, sep. 2012.

[54] A. A. Nacci, V. Rana, in D. Sciuto, „A Perspective Vision on Complex Residential Building Management Systems“, 2014, str. 209–214.

[55] A. A. Nacci, G. Bettinazzi, C. Pilato, V. Rana, M. D. Santambrogio, in D. Sciuto, „A SystemC-based framework for the simulation of appliances networks in energy-aware smart spaces“, 2014, str. 485–490.

[56] Nacci, Alessandro, „WF IoT2014 DAY3 A SystemC Based Framework for the Simulation of Appliances Networks in Energy Aware“, 10-mar-2014.

[57] „Box Project - S-Cube, NECST - Politecnico di Milano, Telecom Italia“. [Na spletu]. Dostopno: http://box.necst.it/index.html.

[58] „Overview — SimPy 3.0.8 documentation“. [Na spletu]. Dostopno: http://simpy.readthedocs.org/en/stable/index.html.

[59] L. G. H. T, W. –. Chang, S. Ha, in E. A. Lee, *Heterogenous Simulation — Mixing Discrete-Event Models with Dataflow*. 1996.

[60] Nayyar, Anand in Singh, Rajeshwar, „A Comprehensive Review of Simulation Tools for Wireless Sensor Networks (WSNs)“.

[61] A. Kroeller, D. Pfisterer, C. Buschmann, S. P. Fekete, in S. Fischer, „Shawn: A new approach to simulating wireless sensor networks“, *arXiv:cs/0502003*, feb. 2005.

[62] Ghica, Oliviu C., „SIDnet-SWANS manual.dvi - SIDnet-SWANS manual.pdf“, 03-mar-2010. [Na spletu]. Dostopno: http://users.eecs.northwestern.edu/~ocg474/SIDnet/SIDnet-SWANS%20manual.pdf.

[63] O. C. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof, in N. Valtchanov, „SIDnet-SWANS: a simulator and integrated development platform for sensor networks applications“, 2008, str. 385.

[64] M. H. Kabir, S. Islam, M. J. Hossain, in S. Hossain, „Detail Comparison of Network Simulators“, *ResearchGate*, let. 5, št. 10, str. 203, nov. 2014.

[65] S. P. Lau, G. V. Merrett, in N. M. White, „Energy-efficient street lighting through embedded adaptive intelligence“, *DeepDyve*, maj 2013.

[66] Z. O. Vilen Looga, „MAMMOTH: A massive-scale emulation platform for Internet of Things“, let. 3, str. 1235–1239, 2012.

[67] B. P. Zeigler, „DEVS today: recent advances in discrete event-based information technology“, str. 148–161, 2003.

[68] Sehili, Souhila, Capocchi, Laurent, in Santucci, Jean-Francois, „IoT Component Design and Implementation using Discrete Event Specification Simulations (PDF Download Available)“. [Na spletu]. Dostopno: https://www.researchgate.net/publication/267029495_IoT_Component_Design_and_Implementation_using_Discrete_Event_Specification_Simulations.

[69] S. Sehili, L. Capocchi, J. F. Santucci, S. Lavirotte, in J. Y. Tigli, „Discrete Event Modeling and Simulation for IoT Efficient Design Combining WComp and DEVSimPy Framework“:, 2015, str. 44–52.

[70] „What is the Smart Grid?“ [Na spletu]. Dostopno: https://www.smartgrid.gov/the_smart_grid/smart_grid.html.

[71] „Overview — mosaik 2.2.0 documentation“. [Na spletu]. Dostopno: https://mosaik.readthedocs.org/en/latest/overview.html.

[72] J. Dede, K. Kuladinithi, A. Förster, O. Nannen, in S. Lehnhoff, „OMNeT++ and mosaik: Enabling Simulation of Smart Grid Communications“, v *ResearchGate*, 2015.

[73] „MESCOS- A Multi-Energy Co-Simulator for City District Energy Systems - RWTH AACHEN UNIVERSITY Institute for Automation of Complex Power Systems - English“. [Na spletu]. Dostopno: http://www.acs.eonerc.rwth-aachen.de/cms/E-ON-ERC-ACS/Forschung/Projekte/~fbfr/MESCOS-Ein-Multi-Energie-Co-Simulator-f/lidx/1/.

[74] „NetLogo“, *Wikipedia, the free encyclopedia*. 19-apr-2016.

[75] „Demand/Activity-based_Demand_Generation - SUMO - Simulation of Urban Mobility“. [Na spletu]. Dostopno: http://sumo.dlr.de/userdoc/Demand/Activity-based_Demand_Generation.html.

[76] „SUMO – Simulation of Urban MObility“. [Na spletu]. Dostopno: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/.

[77] „Introducing DPWS“. [Na spletu]. Dostopno: https://msdn.microsoft.com/en-us/library/dd170125.aspx.

[78] „Devices Profile for Web Services | Web Services for Devices“. [Na spletu]. Dostopno: http://ws4d.org/technology/dpws/.

[79] N. S. Han, „Semantic service provisioning for 6LoWPAN : powering internet of things applications on Web“, *ResearchGate*, jul. 2015.

[80] „C# Tool to Simulate IOT Sensor Telemetry for OpenTSDB“, *Hanuk's Microsoft IoT Strategy Blog*. [Na spletu]. Dostopno: https://blogs.msdn.microsoft.com/hanuk/2015/04/21/c-tool-to-simulate-iot-sensor-telemetry-for-opentsdb/.

[81] „Simulavr - user manual“. [Na spletu]. Dostopno: http://nongnu.askapache.com/simulavr/manual-1.0.pdf.

[82] P. M. Wightman in M. A. Labrador, „Topology Maintenance: Extending the Lifetime of Wireless Sensor Networks“, *IEEE Lat. Am. Trans.*, let. 8, št. 4, str. 469–475, avg. 2010.

[83] Matteo Cesana, „IoT, May 7“, 07-maj-2015. [Na spletu]. Dostopno: https://www.youtube.com/watch?v=oSrT4oOtjXw.

[84] Bagula, Ba in Erasmus, Zenville, „IoT Emulation with Cooja“. [Na spletu]. Dostopno: http://wireless.ictp.it/school_2015/presentations/firstweek/ICTP-Cooja-Presentation-version0.pdf.

[85] R. Khan, „IoT (Internet of Things) : Simulating RIOT-OS applications in Cooja simulator“, *Raees Khan*, 30-apr-2015. [Na spletu]. Dostopno: https://raaees.wordpress.com/2015/04/30/iot-internet-of-things-simulating-riot-os-applications-in-cooja-simulator/.

[86] Baccelli, Emmanuel in Adjih, Cedric, „RIOT“. [Na spletu]. Dostopno: http://www.systematic-paris-

region.org/sites/default/files/content/actualites/attachments/%282015-03%29InnovationSpring-RIOT-v3.pdf.

[87] „TOSSIM - TinyOS Wiki“. [Na spletu]. Dostopno: http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM.

[88] S. Saginbekov in C. Shakenov, „Testing Wireless Sensor Networks with Hybrid Simulators“, *ArXiv160201567 Cs*, feb. 2016.

[89] M. R. H. Khan, R. Passerone, in D. Macii, „FZepel: RF-level power consumption measurement (RF-PM) for Zigbee wireless sensor network-towards cross layer optimization“, 2008, str. 959–966.

[90] Kraemer, Frank Alexander in Herrmann, Peter, „Creating Internet of Things Applications from Building Blocks“, *ERCIM News*, št. 101, apr. 2015.

[91] „Evothings Studio Starter Guide“. [Na spletu]. Dostopno: https://evothings.com/doc/starter-guides/evothings-studio-starter-guide.html.

[92] „IoT Architecture in Simulation“, *wot.io Labs*, 06-okt-2015. [Na spletu]. Dostopno: http://labs.wot.io/iot-architecture-simulation-with-wot-ios-ripple/.

[93] „.NET Gadgeteer - Microsoft Research“. [Na spletu]. Dostopno: http://research.microsoft.com/en-us/projects/gadgeteer/.

[94] Miori, Vittorio in Russo, Dario, „Home automation devices belong to the IoT world (PDF Download Available)“, *ERCIM NEWS*, št. 101, apr. 2015.

[95] W. Xu, „Modeling and exploiting the knowledge base of web of things“, jan. 2015.

[96] M. Samman, S. Khattab, in R. Bahgat, „An Integrated Testbed Environment for the Web of Things“, *ResearchGate*, jun. 2015.

[97] J. Lopes, R. Souza, C. Geyer, C. Costa, J. Barbosa, A. Pernas, in A. Yamin, „A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing“, *J. Univers. Comput. Sci.*, let. 20, št. 9, sep. 2014.

[98] J. Bruneau, „Developing and Testing Pervasive Computing Applications: A Tool-Based Methodology - THESIS DEFENSE“.

[99] „Simulating with DiaSim“, 08-jul-2009. [Na spletu]. Dostopno: https://www.youtube.com/watch?v=T5IhF-JVma0.

[100] D. Cassou, J. Bruneau, C. Consel, in E. Balland, „Toward a Tool-Based Development Methodology for Pervasive Computing Applications“, *IEEE Trans. Softw. Eng.*, let. 38, št. 6, str. 1445–1463, nov. 2012.

[101] „IoT -- Orchestrating Masses of Sensors“. [Na spletu]. Dostopno: http://phoenix.inria.fr/research-projects/iot.